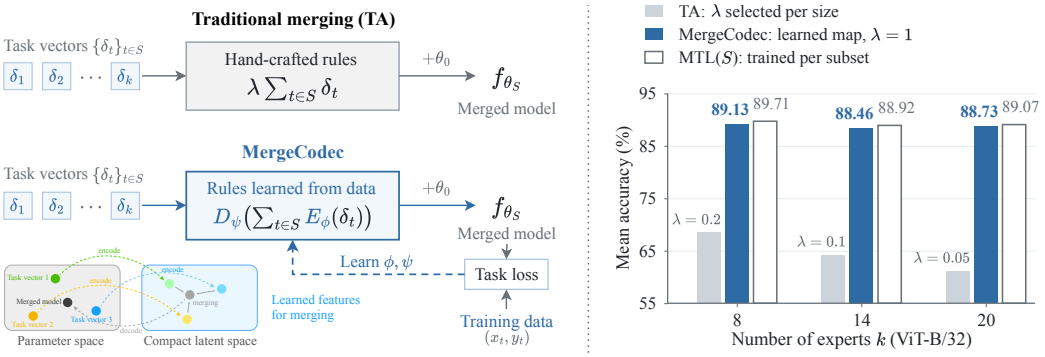
MERGECODEC: LEARNING TO COMPRESS AND COMPOSE TASK VECTORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging aims to combine experts derived from a shared pretrained model, and yet preserving their capabilities across varying task compositions and numbers of experts remains challenging. Existing methods typically rely on hand-crafted merging rules or combination-specific coefficient optimization. We introduce MergeCodec, which learns a reusable merging map through task supervision. MergeCodec encodes task vectors into compact latent representations, composes selected experts by summing their codes, and decodes the result into a merged model update. Training end-to-end over subsets with varying task identities and cardinalities encourages the latent space to support robust composition. Once trained, the encoder can be reused for arbitrary expert combinations without additional task data, gradient updates, or per-combination calibration. Across 20 tasks and 3 CLIP backbones, a single MergeCodec per backbone successfully merges any number of experts, achieving performance comparable to multi-task models trained separately for each subset. Despite never being trained on single subsets, MergeCodec also largely preserves the accuracy of individual experts. Moreover, MergeCodec generalizes to out-of-distribution settings, successfully merging experts unseen during training, and its generalization improves as the training expert bank expands. These results demonstrate that model merging can be learned as a reusable compositional mapping, yielding task-vector representations that generalize across both novel expert combinations and previously unseen experts.



(a) From hand-crafted rules to data-driven learned rules

(b) Model merging performance

Figure 1: **Learn to merge, then reuse.** TA illustrates a hand-crafted rule, while MergeCodec learns a reusable mapping through supervised training. TA's coefficients are selected on standard-set test scores for each subset.

1 INTRODUCTION

Model merging combines the capabilities of task-specific experts into a single model (Yang et al., 2026a; Cai et al., 2026). Experts fine-tuned from a common base can be represented by task vectors, their parameter differences from that base (Ilharco et al., 2023), and assembled into models for selected tasks (Wang et al., 2026). Applications range from improving accuracy and robustness without ensemble inference costs (Wortsman et al., 2022) to combining specialized language-model capabilities in a single model (Yu et al., 2024).

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.

Many model-merging methods prescribe how task vectors should be combined through hand-designed rules, ranging from scaled addition (Ilharco et al., 2023) to low-rank and singular-vector constructions (Gargiulo et al., 2025). Adaptive methods optimize coefficients using unlabeled images (Yang et al., 2024) or optimize updates using task-vector guidance without additional data (Cheng et al., 2025).

However, manually designed merging rules or objectives often involve tuning or calibration, while substantial gaps to multi-task learning (MTL) can remain (Table 1). We therefore ask:

> *Can a shared module learn to compose the task vectors it receives, so that it can be reused across combinations without further tuning or calibration after training?*

Figure 1 illustrates this goal: TA uses different selected coefficients across subset sizes, whereas MergeCodec approaches separately trained MTL models with one fixed coefficient. Reusable merging is also explored by HyperFix, which predicts corrections to a linear merge (Kim & Wang, 2026); our approach learns compact representations for composition.

We introduce **MergeCodec**, an encoder-decoder that compresses individual task vectors, sums their codes, and decodes a model update. We retain simple addition in latent space and learn the representations and decoding map that make the supplied updates work together. The sum is invariant to expert order. We train the codec end to end with labeled task losses over subsets of varying membership and size. After training, the frozen codec uses $\lambda = 1$ and requires no additional task examples, gradient updates, or coefficient search for new expert sets. It produces a static model without adding computation to per-example inference.

We evaluate MergeCodec with three CLIP backbones (Radford et al., 2021) under FusionBench settings (Tang et al., 2025). On in-domain tasks, a single codec maintains high performance across task combinations and subset sizes unseen during training, approaching separately trained MTL models. This generalization extends even to single-expert reconstruction: the codec closely preserves expert accuracy despite never being trained on singleton inputs. In out-of-distribution (OOD) settings, it further generalizes to experts whose tasks are excluded from both initialization and training, achieving performance competitive with strong merging baselines. Together, these results suggest that the codec learns a reusable ability to compose task vectors, extending across new combinations and previously unseen tasks. Our contributions are:

- **A codec for model merging.** We design a trainable encoder-decoder that compresses task vectors and composes expert sets through latent addition.
- **Mixed training for stable merging.** Supervised training across diverse subsets enables stable, high performance with fixed $\lambda = 1$, without per-combination calibration.
- **Extensive experimental validation.** Experiments cover subset merging, expert recovery, unseen-expert transfer, and ablations of codec architecture and training.

2 Related Work

Model Merging. Existing methods combine checkpoints or task vectors through averaging and scaled addition (Wortsman et al., 2022; Ilharco et al., 2023), and address update interference through sparsification, sign resolution, geometric constraints, or functional alignment (Yadav et al., 2023; Yu et al., 2024; Yang et al., 2026b; Shi et al., 2025). Their dependence on parameter structure also motivates defenses that inhibit unauthorized merging while preserving model functionality (Wang et al., 2025; Wei et al., 2025a).

Adaptive approaches optimize coefficients or feature corrections for a selected merge using data (Yang et al., 2024; Wei et al., 2025b; Gu et al., 2026), or optimize updates using task-vector guidance without additional examples (Cheng et al., 2025). Learning can also be reused across combinations: HyperFix predicts subset-conditioned corrections to a linear merge with a shared hypernetwork (Kim & Wang, 2026). MergeCodec learns compact codes whose sum is decoded into a merged update. Training across diverse subsets enables frozen reuse with a fixed coefficient and no per-combination optimization.

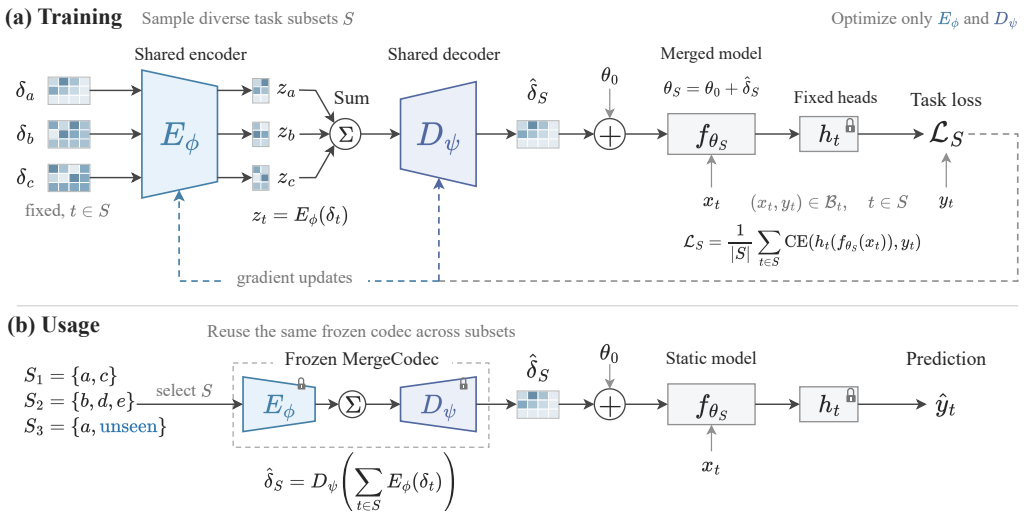


Figure 2: **MergeCodec training and inference.** (a) Supervised learning of task-vector composition. (b) Frozen composition with $\lambda = 1$. Uncompressed parameters use direct summation.

Latent-space Merging. Parameter representations support the prediction of model properties and the construction of models from latent codes (Schürholt et al., 2021; Soro et al., 2026). Low-rank approximations and shared bases further expose compact structure in task vectors for merging and arithmetic (Gargiulo et al., 2025; Zeng et al., 2026). MergeCodec learns tensor-wise representations for experts from a common base through the supervised task losses of composed models, without a parameter-reconstruction objective. Its encode-sum-decode structure follows the permutation-invariant aggregation pattern of Deep Sets (Zaheer et al., 2017), while training on diverse subsets shapes the latent space for variable-size composition.

3 MergeCodec: Model Merging via Compact Representations

Let $\theta_0 \in \mathbb{R}^P$ denote a common pretrained model and θ_t a task-specific expert with the same architecture and parameter ordering. Its task vector is $\delta_t = \theta_t - \theta_0$ (Ilharco et al., 2023). For a nonempty expert set S, merging constructs a single feature extractor f_{θ_S} used with fixed task-specific heads h_t. Task Arithmetic adds scaled updates, $\theta_S^{\mathrm{TA}} = \theta_0 + \lambda \sum_{t \in S} \delta_t$. We instead learn a composition function M_η for variable-size inputs, $\theta_S = \theta_0 + M_\eta(\{\delta_t : t \in S\})$. Its parameters are trained on an expert bank $\mathcal{T}_{\mathrm{train}}$ using labeled datasets $\{\mathcal{D}_t\}_{t \in \mathcal{T}_{\mathrm{train}}}$, then frozen for reuse across expert sets. The selected task vectors must be available when constructing each merged model.

MergeCodec learns to compose task vectors through supervised task losses, then reuses the frozen codec across expert subsets (Figure 2).

3.1 Learning Composable Task-Vector Representations

An encoder E_ϕ compresses each task vector into a low-dimensional representation, and a decoder D_ψ maps representations from the compact space back to the original parameter space:

$$z_t = E_\phi(\delta_t), \qquad z_S = \sum_{t \in S} z_t, \qquad \theta_S = \theta_0 + D_\psi(z_S). \tag{1}$$

Here $\eta = (\phi, \psi)$ comprises the codec parameters. The sum accepts variable-size expert sets and is invariant to their ordering. Each expert is encoded separately and the aggregate is decoded once, with a fixed outer coefficient $\lambda = 1$ during training and primary evaluation.

Tensor-wise encoding and decoding. Each compressible tensor has its own codec, shared across experts, with rank chosen to meet a target dimension compression factor. Let $X_{t,\ell} \in \mathbb{R}^{m_\ell \times n_\ell}$ denote tensor ℓ's task update, oriented so that $m_\ell \geq n_\ell$. Independent encoder and decoder bases

$A_\ell, B_\ell \in \mathbb{R}^{m_\ell \times r_\ell}$ have learned orthonormal columns and fixed rank $r_\ell \le n_\ell$:

$$Z_{t,\ell} = R^E_\ell(A_\ell^\top X_{t,\ell}), \qquad Z_{t,\ell} \in \mathbb{R}^{r_\ell \times n_\ell}, \tag{2}$$

$$\widehat{X}_{S,\ell} = B_\ell R^D_\ell \left(\sum_{t \in S} Z_{t,\ell} \right). \tag{3}$$

The bias-free residual maps R^E_ℓ and R^D_ℓ apply learned channel-wise scaling, RMS normalization, and GELU independently to each latent column. They preserve zero; applying R^D_ℓ after summation makes the decoded update nonlinear in the selected combination. Scalar and vector parameters use identity maps, so their updates are summed directly. We define compression as $C = P/d_{\text{code}}$, where d_{code} includes these uncompressed coordinates but excludes shared codec parameters. Appendix B.1 specifies the residual maps, basis parameterization, and parameter counts.

Low-rank task approximations can retain task performance (Gargiulo et al., 2025), motivating a bottleneck that may suppress noise and redundancy while supervision preserves useful information. Lower ranks also reduce the sizes of the learned bases and residual maps, keeping the codec compact.

SVD-Based Initialization. For each tensor, we initialize A_ℓ and B_ℓ as independent copies of the leading left singular vectors of the concatenated training updates $H_\ell = [X_{t,\ell}]_{t \in \mathcal{T}_{\text{train}}}$. The residual maps start as identities. Both bases subsequently evolve independently, while ranks remain fixed. Rank allocation and initialization use only the training bank, including in unseen-expert experiments. SVD supplies a data-dependent starting point; random orthonormal bases are also compatible with the method. Appendix B.1 gives the construction details.

3.2 End-to-End Supervision with Mixed Task Subsets

Each episode selects an expert subset $S \subseteq \mathcal{T}_{\text{train}}$ and a labeled minibatch $\mathcal{B}_t \subset \mathcal{D}_t$ for each $t \in S$. Using the merged model from Eq. (1), we minimize

$$\mathcal{L}_S(\phi, \psi) = \frac{1}{|S|} \sum_{t \in S} \frac{1}{|\mathcal{B}_t|} \sum_{(x,y) \in \mathcal{B}_t} \text{CE}(h_t(f_{\theta_S}(x)), y). \tag{4}$$

Tasks receive equal weight within each episode. Gradients pass through the merged model and codec, updating only (ϕ, ψ); pretrained weights, task vectors, heads, and ranks stay fixed. Training uses no parameter-reconstruction or distillation loss.

Mixed sampling. We vary both subset membership and cardinality throughout training, randomly interleaving a fixed quota of episodes at each size with balanced expert exposure. Each selected task supplies labeled images from its own training stream. Reserved non-full evaluation combinations are excluded; the full-bank combination is shared between training and evaluation. This schedule trains one codec to serve diverse combinations with the same $\lambda = 1$, rather than optimizing only the complete bank.

Reuse after training. Expert codes can be cached, or newly supplied task vectors encoded by the frozen maps. Summing and decoding the selected codes requires no task examples or gradient updates. Adding the update to θ_0 materializes a static model, with no codec computation during per-example inference.

4 Experiments and Results

4.1 Experimental Setup

We follow the CLIP model-fusion settings of FusionBench (Tang et al., 2025) to test whether a frozen codec can compose the task vectors it receives. We evaluate new subsets of training-bank experts, single-expert reconstruction, and transfer to experts excluded from codec training.

4.1.1 CLIP Image Classification

We use CLIP (Radford et al., 2021) ViT-B/32, ViT-B/16, and ViT-L/14 backbones (Dosovitskiy et al., 2021), each with 20 experts from a common pretrained model and fixed task-specific text

heads. We follow FusionBench's complete train/test splits without an additional codec validation split; datasets and subset membership appear in Appendix A. Unless stated otherwise, one codec per backbone trains on the 20-expert bank for 4,000 Mixed updates at $C = 4$. We evaluate the final checkpoint; optimizer and sampling details are in Appendix B.

4.1.2 Compared Methods

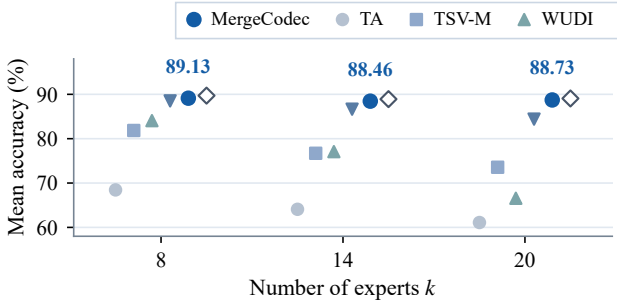
On ViT-B/32 we compare **TA** (Ilharco et al., 2023), **TSV-M** (Gargiulo et al., 2025), **WUDI** (Cheng et al., 2025), and **WUDI+ProbSurgery** (Wei et al., 2025b); larger backbones include TA and supervised references. **Experts** evaluates each expert on its own task; **MTL**(S) trains a model from the common base for each subset S, with fixed heads. MTL and MergeCodec both use labeled training images, but MergeCodec is trained once per backbone. Appendix C also reports Base and Full20-only.

4.1.3 Calibration Setup

For TA and TSV-M, we select one outer merge coefficient on each standard set's test accuracy and reuse it for random sets of the same size; these baselines are *test-informed*. WUDI optimizes each merge using task vectors; ProbSurgery additionally adapts each merge on unlabeled test images. Neither uses an extra outer coefficient. MergeCodec fixes $\lambda = 1$ in training and evaluation, using frozen maps to combine new subsets without further tuning, data, or gradient updates. Its static model needs no task-specific correction at inference. Appendix B gives the complete protocol.

4.2 Multi-Task Subset Merging

We evaluate the standard and four random sets at $k = 8, 14$, plus the full 20-task bank. All experts were seen in training, but the ten non-full combinations and cardinality $k = 14$ were not. Scores average tasks equally within each set, then the five sets at $k = 8, 14$. Each backbone uses one frozen codec with $\lambda = 1$ for all sets.



(a) Accuracy across subset sizes.

Method	λ selection	Extra images	Gradient updates
TA†	Search	Labeled	None
TSV-M†	Search	Labeled	None
WUDI	None	None	Yes
WUDI+PS	None	Unlabeled	Yes
Codec	**Fixed 1**	**None**	**None**

(b) Requirements for reuse.

Figure 3: **One trained codec, reused across expert subsets.** ViT-B/32; $k = 8, 14$: five-set means; $k = 20$: full bank. MTL(S) is trained separately per subset. The right panel concerns merging after codec training. †: coefficient search uses labeled test images in our protocol; PS: ProbSurgery.

One codec serves new combinations and subset sizes. MergeCodec remains close to separately trained MTL(S) models without per-combination calibration (Figure 3). Its ViT-B/32 gains over WUDI+ProbSurgery are 0.61, 1.81, and 4.32 points at $k = 8, 14, 20$ (Table 1), with gaps to MTL(S) below 0.6 points. It wins on ten of 11 combinations, trailing by only 0.04 points on 8-R2 (Table 11). Thus the same learned mapping supports both new task combinations and the unseen cardinality $k = 14$.

Reuse extends to larger backbones. ViT-B/16 and ViT-L/14 also remain within 0.27-0.73 points of independently trained MTL(S), using one codec and fixed $\lambda = 1$ per backbone (Table 2). Table 12 gives complete subset scores.

k	Experts	MTL(S)	TA†	TSV-M†	WUDI	WUDI+PS‡	MergeCodec
8	90.15	89.71	68.45	81.85	84.07	88.52	**89.13**$_{+0.61}$
14	89.51	88.92	64.08	76.71	77.08	86.65	**88.46**$_{+1.81}$
20	89.77	89.07	61.10	73.56	66.57	84.41	**88.73**$_{+4.32}$

Table 1: **ViT-B/32 subset merging (accuracy, %).** $k = 8, 14$: five-set means; $k = 20$: full bank. Bold: best merging method. Subscripts: gain over WUDI+PS (pp). †: test-selected coefficients; ‡: unlabeled test-time adaptation; PS: ProbSurgery. Individual sets appear in Table 11.

	ViT-B/16					ViT-L/14				
k	Experts	MTL(S)	TA†	Codec	Δ_{MTL}	k	Experts	MTL(S)	TA†	Codec	Δ_{MTL}
8	92.14	92.02	72.54	91.30	-0.73	8	93.99	94.01	78.46	93.55	-0.46
14	91.46	91.30	67.95	90.75	-0.55	14	93.39	93.44	74.12	92.92	-0.53
20	91.62	91.35	64.81	91.08	-0.27	20	93.53	93.60	71.14	92.98	-0.62

Table 2: **Larger-backbone subset merging (accuracy, %).** $k = 8, 14$: five-set means. Δ_{MTL}: Codec minus MTL(S), in percentage points. †: test-selected coefficients.

Individual expert capabilities are preserved. Despite receiving no singleton training episodes, the frozen codec reconstructs all 20 individual experts with mean accuracy losses of only 0.26, 0.21, and 0.14 points on ViT-B/32, ViT-B/16, and ViT-L/14 (Figure 4). Thus reuse also extends to single inputs; per-task scores appear in Table 13.

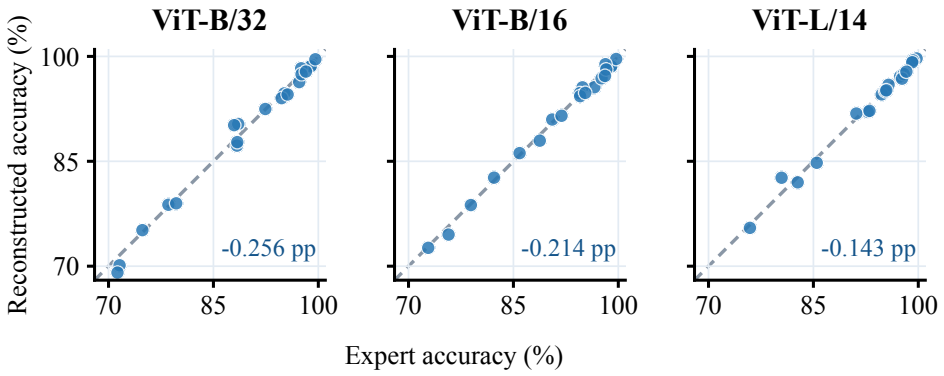


Figure 4: **Single-expert reconstruction.** One point per task; dashed line: equal accuracy. Insets: mean accuracy gap (reconstructed minus expert, pp).

4.3 Generalization to Unseen Experts

Protocol. We train ViT-B/32 codecs on nested banks of $n \in \{4, 8, 12, 16\}$ experts, excluding SVHN, Flowers102, PCAM, and Rendered-SST2 from both initialization and training. Their task vectors are supplied only afterward, without target-task adaptation. The same four singleton inputs, six pairs, and complete four-expert set are evaluated at every n, with task-balanced averaging. TA and TSV-M select coefficients using target test labels, sharing one coefficient per k across combinations and all n; Codec fixes $\lambda = 1$. WUDI uses no outer scaling or target images. Appendix E gives training, calibration, and evaluation details.

Learned composition transfers to unseen experts. At $n = 16$, four-expert merging reaches 83.15% mean accuracy versus 54.31% for Base, improving every held-out task by 12.33–62.81 points. It also exceeds TSV-M (81.62%), WUDI (81.52%), and TA (77.71%), outperforming WUDI on three of four tasks (Table 17). Two-expert accuracy is within 0.09 points of WUDI. These results show that the frozen codec can combine capabilities from experts whose tasks were absent from its initialization and training.

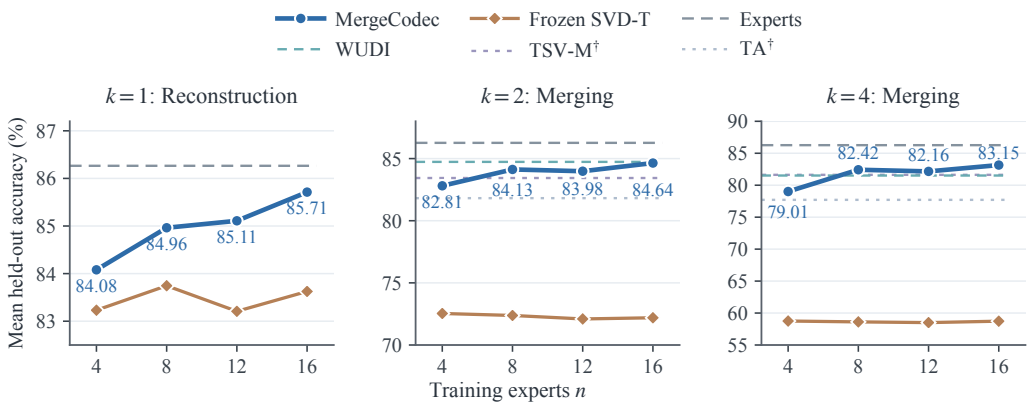
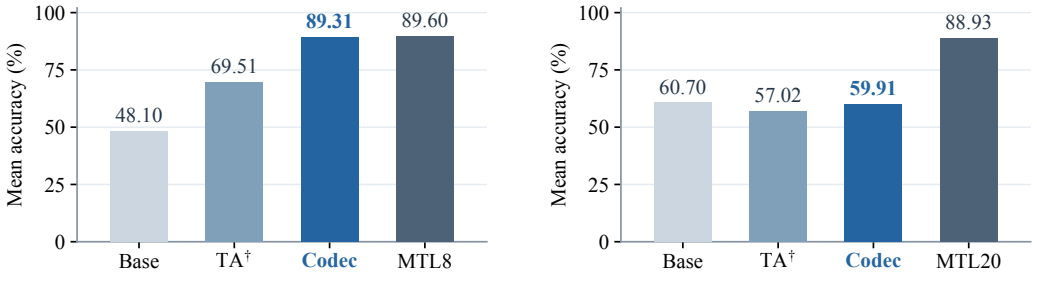


Figure 5: **Unseen-expert reconstruction and merging on ViT-B/32.** $k = 1$: single-expert reconstruction; $k = 2, 4$: merging. Codec uses $\lambda = 1$. Panels use separate vertical scales. †: target-test-selected coefficients.

A broader training bank improves transfer. Expanding the bank from four to 16 experts raises two- and four-expert accuracy by 1.83 and 4.14 points, with particularly strong gains on Flowers102 and PCAM, although the trend is not monotonic (Figure 5). Singleton reconstruction likewise rises from 84.08% to 85.71%, only 0.56 points below the original experts. At $n = 16$, the untrained Frozen SVD-T control reaches 83.63% in reconstruction and 58.73% in four-expert merging, supporting the benefit of learning beyond initialization. Appendix E gives per-task results and seen/unseen mixtures (Table 18).

4.4 Composition or Task Knowledge?

To test whether performance mainly reflects stored training-task knowledge, we give the 20-task ViT-B/32 codec only the eight CLIP8 vectors, then evaluate those tasks and the other 12. All were seen during codec training. Figure 6 compares Base, CLIP8-calibrated TA, and Codec, with MTL8/20 as supervised references for input/omitted tasks.



(a) Eight tasks included in the merge.

(b) Twelve tasks omitted from the merge.

Figure 6: **Composition or task knowledge?** ViT-B/32; Codec and TA each reuse one CLIP8-merged model across panels. MTL8/20 are supervised references. †: $\lambda = 0.2$ selected on CLIP8.

High performance follows the supplied task vectors. Codec reaches 89.31% on input tasks, close to MTL8 (89.60%). On omitted tasks, its 59.91% is comparable to Base (60.70%) and TA (57.02%), well below MTL20 (88.93%). Training on these tasks therefore does not automatically improve their average accuracy when their vectors are absent. Together with OOD transfer, this supports learning to compose supplied task vectors rather than broadly injecting training-task knowledge. Appendix F.3 gives details.

4.5 Ablation Studies

All ablations use ViT-B/32, 4,000 updates, and $\lambda = 1$, evaluated on 20 singleton inputs and the 11 multi-task sets. Unless varied, $C = 4$. Matched controls are detailed in Appendix B. Tables 3 and 4 mark each accuracy column's best score in bold and second-best with underlining.

Variant	$k = 1$	$k = 8$	$k = 14$	$k = 20$
Linear	<u>89.68</u>	88.60	87.68	88.02
Encoder-only	**89.82**	**89.31**	<u>88.36</u>	<u>88.36</u>
Two-sided	89.51	<u>89.13</u>	**88.46**	**88.73**

Table 3: **Codec structure (accuracy, %).**

Variant	$k = 1$	$k = 8$	$k = 14$	$k = 20$
Full20-only	84.98	85.06	86.54	**88.81**
Curriculum	<u>86.32</u>	<u>88.50</u>	<u>88.38</u>	88.57
Mixed	**89.51**	**89.13**	**88.46**	<u>88.73</u>

Table 4: **Subset sampling (accuracy, %).**

Nonlinearity benefits composition beyond reconstruction. Two-sided nonlinearity improves over the linear codec by 0.53-0.78 points at $k = 8, 14, 20$ (Table 3). Encoder-only performs better on singleton and eight-task inputs but worse at $k = 14, 20$, showing that better individual reconstruction need not yield better composition. The nonlinear maps add very few parameters relative to the learned bases (Appendix B.1), making their use in both the encoder and decoder a favorable tradeoff between parameter count and merging performance.

Mixed subsets enable reuse beyond the full bank. At $\lambda = 1$, Mixed outperforms Curriculum at every evaluated size and improves over Full20-only by 1.92-4.53 points at $k = 1, 8, 14$, with only a 0.08-point decrease at $k = 20$ (Table 4). These results support Mixed as our default: it maintains strong performance across subset sizes with a fixed coefficient while largely preserving full-bank performance. Curriculum orders the same episodes by increasing cardinality; Full20-only always uses the complete bank, with greater per-task exposure (Appendix B).

Compression factor. Table 5 varies C with matched Mixed episodes and image batches. Relative to $C = 4$, $C = 2$ gains 0.17-0.23 points at $k = 8, 14, 20$, while $C = 8$ loses 0.31-0.40 points and reduces shared parameters from 114.75M to 60.63M. The default $C = 4$ balances capacity and accuracy; storage details appear in Appendix B.3.

C	Params	$k = 1$	$k = 8$	$k = 14$	$k = 20$
2	174.89	89.87	89.31	88.63	88.96
4	114.75	89.51	89.13	88.46	88.73
8	60.63	89.07	88.73	88.10	88.42

Table 5: **Target compression (accuracy, %).** Shared codec parameters in millions.

Initialization. Figure 7 compares SVD with random orthogonal bases at the same SVD-derived ranks. SVD yields lower middle/late training CE (final 100-step means: 0.084 versus 0.094), yet final accuracies differ by at most 0.33 points at the displayed precision. SVD improves training fit, but random orthogonal initialization also supports strong final performance.

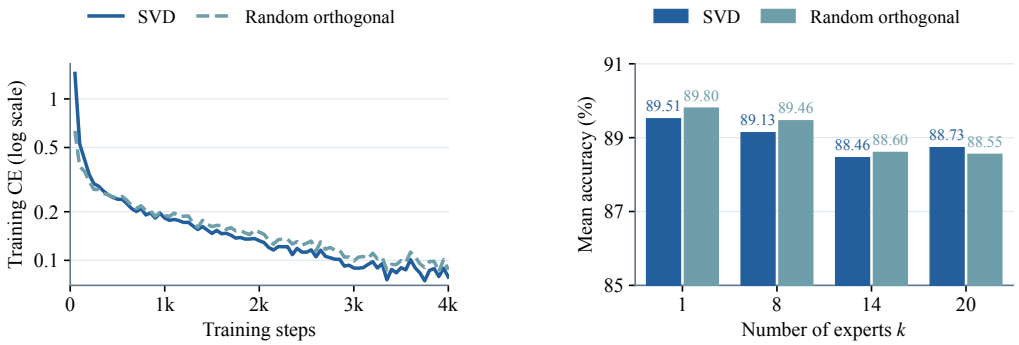


Figure 7: **Initialization ablation on ViT-B/32.** Left: 50-step mean training CE on a log scale. Right: test accuracy after 4,000 updates, with $\lambda = 1$.

4.6 EFFICIENCY

Training once enables inexpensive repeated merging. Codec incurs an upfront training cost: updates take 1.86-2.41 times as long as MTL20 across backbones, despite sampling 12 tasks on average versus 20 (Table 6). These BF16 profiles measure 100 updates after 10 warmup steps on one H100 80GB, excluding batch waiting. The resulting codec is then reused across expert combinations.

After training, cached-code merging takes 18-33 ms on ViT-B/32, versus 10-23 ms for TA; encoding fresh inputs raises this to 87-193 ms. TSV-M and WUDI take tens of seconds, and WUDI+PS hundreds, in the measured implementations. FP32 timings exclude training, loading, coefficient search, and preparation: Codec additionally requires QR export (about 0.35 s) and cached-code construction. PS reuses expert features, while TSV-M uses its full decomposition implementation. Protocols and preparation costs appear in Appendix F. The measured benefit is low merge-time cost after training; the overall benefit depends on how often the codec is reused.

Training

Backbone	MTL20	Codec	Ratio
ViT-B/32	0.410	0.989	2.41×
ViT-B/16	0.479	1.023	2.14×
ViT-L/14	1.378	2.565	1.86×

Merging

Method	$k = 8$	$k = 14$	$k = 20$
TA	0.010	0.017	0.023
Codec (cached)	0.018	0.025	0.033
Codec (encode)	0.087	0.135	0.193
TSV-M	27.086	43.965	60.474
WUDI	50.855	82.059	113.985
WUDI+PS	248.572	375.126	498.450

Table 6: **Efficiency on one H100 (seconds).** Training: mean per update, BF16. Merging: median, ViT-B/32 FP32. Preparation excluded.

5 LIMITATIONS

Despite its strong merging performance, MergeCodec incurs a substantial upfront training cost, which remains a practical bottleneck. Unlike merging rules that operate directly on expert parameters, it requires supervised codec training; in our profiles, each update costs 1.86–2.41 times as much as MTL20 (Table 6). This cost is incurred once for a given codec and can be amortized by reusing it across expert combinations, although the benefit depends on the number of subsequent merges. Reducing this training cost while preserving composition quality and reusability is an important direction for future work.

MergeCodec's OOD results suggest promising generalization potential: with 16 training experts, it approaches WUDI on two-expert merges and surpasses it on four-expert merges, without target-task adaptation (Figure 5). Performance improves overall as the training bank expands, although a gap to a codec trained on all target tasks remains (Appendix E). Expanding the diversity of training experts and evaluating broader task and model families are promising next steps toward a more generally reusable codec.

6 CONCLUSION

We introduced MergeCodec, a trainable encoder–decoder that learns compact task-vector representations for composition. Mixed subset training enables one frozen codec to combine varying expert sets with a fixed coefficient, without additional data or per-combination calibration. Experiments across three CLIP backbones show performance close to separately trained multi-task models while largely preserving individual expert capabilities. Transfer to experts excluded from initialization and training further supports the reuse of learned composition across new inputs. These findings establish learning composable representations as a promising direction for model merging. Future work can reduce the upfront training cost and evaluate this approach across broader model families and task domains.

AI USE STATEMENT

We used generative AI tools to assist with code implementation, evaluation, result aggregation, and manuscript drafting. The authors designed and led the experiments and take full responsibility for the reported results and final manuscript.

REPRODUCIBILITY STATEMENT

Section 3 defines the codec and its training objective. Appendix B.1 specifies the residual maps, basis parameterization, rank allocation, and initialization. Appendices A and B document dataset splits, evaluation subsets, optimization, precision settings, random seeds, and baseline calibration policies, including the use of test labels for coefficient selection. Complete subset and single-expert results appear in Appendices C and D; Appendix E specifies the held-out expert protocol. Appendix F describes timing boundaries and the omitted-task diagnostic. These detailed records of hyperparameters and experimental configurations provide a strong foundation for reproducing our experiments.

ETHICS STATEMENT

This work studies model merging using existing image-classification benchmarks and pretrained models. It does not introduce a new dataset or involve recruiting participants. The benchmarks include facial-expression and medical-image tasks; their use here evaluates classification performance and does not establish suitability for decisions about individuals or clinical use. Merging may preserve or change biases and sensitive information inherited from the base model and experts. Our accuracy evaluations do not assess fairness, privacy, or deployment safety. Use of merged models should respect the source models' and datasets' licenses and usage restrictions, and deployment requires assessments appropriate to the intended application. The training costs reported in this work also motivate efforts to reduce its computational resource use.

REFERENCES

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pp. 446–461. Springer, 2014. doi: 10.1007/978-3-319-10599-4_29. URL https://link.springer.com/chapter/10.1007/978-3-319-10599-4_29. 16

Shuo Cai, Yanggan Gu, Zihao Wang, Yuanyi Wang, Yibo Yan, Wenjun Wang, Yuhang Liu, Guanghao Zhu, Sirui Huang, Ming Li, and Hongxia Yang. From parameters to behaviors: A survey of model fusion for large language models. *arXiv preprint arXiv:2609.19553*, 2026. URL https://arxiv.org/abs/2609.19553. 1

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. doi: 10.1109/JPROC.2017.2675998. URL https://doi.org/10.1109/JPROC.2017.2675998. 16

Runxi Cheng, Feng Xiong, Yongxian Wei, Wanyun Zhu, and Chun Yuan. Whoever started the interference should end it: Guiding data-free model merging via task vectors. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10121–10143. PMLR, 2025. URL https://proceedings.mlr.press/v267/cheng25h.html. 2, 5

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. doi: 10.1109/CVPR.2014.461. URL https://www.robots.ox.ac.uk/~vgg/publications/2014/Cimpoi14/. 16

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature. In *NeurIPS Workshop on Machine Learning*

for Creativity and Design, 2018. URL https://nips2018creativity.github.io/doc/deep_learning_for_classical_japanese_literature.pdf. 16

Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223. PMLR, 2011. URL https://proceedings.mlr.press/v15/coates11a.html. 16

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: Extending MNIST to handwritten letters. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 2921–2926, 2017. doi: 10.1109/IJCNN.2017.7966217. URL https://doi.org/10.1109/IJCNN.2017.7966217. 16

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy. 4

Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. Task singular vectors: Reducing task interference in model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18695–18705, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Gargiulo_Task_Singular_Vectors_Reducing_Task_Interference_in_Model_Merging_CVPR_2025_paper.html. 2, 3, 4, 5

Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. In *Neural Information Processing*, volume 8228 of *Lecture Notes in Computer Science*, pp. 117–124. Springer, 2013. doi: 10.1007/978-3-642-42051-1_16. URL https://link.springer.com/chapter/10.1007/978-3-642-42051-1_16. 16

Yanggan Gu, Shuo Cai, Zihao Wang, Wenjun Wang, Yuanyi Wang, Pengkai Wang, Sirui Huang, Su Lu, Jianmin Wu, and Hongxia Yang. FeatCal: Feature calibration for post-merging models. *arXiv preprint arXiv:2605.13030*, 2026. doi: 10.48550/arXiv.2605.13030. URL https://arxiv.org/abs/2605.13030. 2

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. doi: 10.1109/JSTARS.2019.2918242. URL https://doi.org/10.1109/JSTARS.2019.2918242. 16

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. doi: 10.48550/arXiv.1606.08415. URL https://arxiv.org/abs/1606.08415. 17

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*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj. 1, 2, 3, 5

Hyo Seo Kim and Ren Wang. HyperFix: Combinatorial nonlinear correction for task vector merging. *arXiv preprint arXiv:2608.11499*, 2026. doi: 10.48550/arXiv.2608.11499. URL https://arxiv.org/abs/2608.11499. 2

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980. 17

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013. doi: 10.1109/ICCVW.2013.77. URL https://cs.stanford.edu/groups/vision/pdf/3drr13.pdf. 16

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. 16

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. doi: 10.1109/5.726791. URL https://bottou.org/papers/lecun-98h. 16

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7. 17

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. URL https://research.google/pubs/reading-digits-in-natural-images-with-unsupervised-feature-learning/. 16

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 722–729, 2008. doi: 10.1109/ICVGIP.2008.47. URL https://www.robots.ox.ac.uk/~vgg/publications/2008/Nilsback08/. 16

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, 2012. doi: 10.1109/CVPR.2012.6248092. URL https://www.robots.ox.ac.uk/~vgg/publications/2012/parkhi12a/. 16

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 *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL https://proceedings.mlr.press/v139/radford21a.html. 2, 4, 16

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In *Advances in Neural Information Processing Systems*, volume 34, pp. 16481–16493. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/89562dccfeb1d0394b9ae7e09544dc70-Abstract.html. 3

Kexuan Shi, Yandong Wen, and Weiyang Liu. Model merging with functional dual anchors. *arXiv preprint arXiv:2510.21223*, 2025. doi: 10.48550/arXiv.2510.21223. URL https://arxiv.org/abs/2510.21223. 2

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1170/. 16

Bedionita Soro, Aoxuan Silvia Zhang, Bruno Andreis, Jaehyeong Jo, Song Chong, and Sung Ju Hwang. LS-Merge: Merging language models in latent space. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ef0f1d04d0c43fd3e8f9fb9e56111d50-Abstract-Conference.html. 3

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 1453–1460, 2011. doi: 10.1109/IJCNN.2011.6033395. URL https://doi.org/10.1109/IJCNN.2011.6033395. 16

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, 2025. URL https://www.jmlr.org/papers/v26/25-1243.html. 2, 4

Bastiaan S. Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, volume 11071 of *Lecture Notes in Computer Science*, pp. 210–218. Springer, 2018. doi: 10.1007/978-3-030-00934-2_24. URL https://link.springer.com/chapter/10.1007/978-3-030-00934-2_24. 16

Yuanyi Wang, Yanggan Gu, Zihao Wang, Kunxi Li, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. MergePipe: A budget-aware parameter management system for scalable LLM merging. *arXiv preprint arXiv:2602.13273*, 2026. doi: 10.48550/arXiv.2602.13273. URL https://arxiv.org/abs/2602.13273. 1

Zihao Wang, Enneng Yang, Lu Yin, Shiwei Liu, and Li Shen. Model unmerging: Making your models unmergeable for secure model sharing. *arXiv preprint arXiv:2509.01548*, 2025. doi: 10.48550/arXiv.2509.01548. URL https://arxiv.org/abs/2509.01548. 2

Junhao Wei, Zhe Yu, and Jun Sakuma. Disrupting model merging: A parameter-level defense without sacrificing accuracy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17698–17707, 2025a. doi: 10.1109/ICCV51701.2025.01644. URL https://openaccess.thecvf.com/content/ICCV2025/html/Junhao_Disrupting_Model_Merging_A_Parameter-Level_Defense_Without_Sacrificing_Accuracy_ICCV_2025_paper.html. 2

Qi Wei, Shuo He, Enneng Yang, Tingcong Liu, Haobo Wang, Lei Feng, and Bo An. Representation surgery in model merging with probabilistic modeling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66015–66032. PMLR, 2025b. URL https://proceedings.mlr.press/v267/wei25c.html. 2, 5

Mitchell Wortsman, Gabriel Ilharco, Samir Ya 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 *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html. 1, 2

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. doi: 10.48550/arXiv.1708.07747. URL https://arxiv.org/abs/1708.07747. 16

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, 2010. doi: 10.1109/CVPR.2010.5539970. URL https://3dvision.princeton.edu/projects/2010/SUN/. 16

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 7093–7115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0310. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html. 2

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, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/62868cc2fc1eb5cdf321d05b4b88510c-Abstract-Conference.html. 2

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in LLMs, MLLMs, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58(8):1–41, 2026a. doi: 10.1145/3787849. URL https://doi.org/10.1145/3787849. Article 216. 1

Sihan Yang, Kexuan Shi, and Weiyang Liu. Orthogonal model merging. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/61531. 2

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57755–57775. PMLR, 2024. URL https://proceedings.mlr.press/v235/yu24p.html. 1, 2

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html. 3

Siqi Zeng, Yifei He, Meitong Liu, Weiqiu You, Yifan Hao, Yao-Hung Hubert Tsai, Makoto Yamada, and Han Zhao. Task vector bases: A unified and scalable framework for compressed task arithmetic. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=zkc7u3mIaE. 3

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, pp. 12381–12392. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html. 17

Appendix

Table of Contents

A Datasets and Evaluation Subsets **16**

B Implementation and Calibration Details **16**

B.1 Codec Architecture and Initialization 16

B.2 Training and Calibration 17

B.3 Compression Factor and Storage 18

C Complete Subset and Reconstruction Results **19**

D Single-Expert Reconstruction Scores **19**

E Unseen-Expert Protocol and Additional Results **20**

F Efficiency and Transfer Beyond the Merge Inputs **22**

F.1 Training Efficiency 22

F.2 Merging Efficiency 23

F.3 Tasks Omitted from the Merge Inputs 23

A Datasets and Evaluation Subsets

Table 7 records the dataset sizes loaded by the FusionBench configurations used in these runs. The three backbones use the same split identities and task definitions, with their respective CLIP preprocessing and fixed text classification heads. The counts describe our loaded configurations and need not equal every dataset's original published partition. Rendered-SST2 uses the image-rendered version introduced with CLIP (Radford et al., 2021), based on binary sentiment examples from the Stanford Sentiment Treebank (Socher et al., 2013).

Table 8 specifies all 11 multi-task combinations used for the main comparisons. Random sets are fixed across methods and backbones. All tasks remain individually available during training in this subset-merging experiment.

ID	Dataset	Classes	Train	Test
1	SUN397 (Xiao et al., 2010)	397	19,850	19,850
2	Cars (Krause et al., 2013)	196	8,144	8,041
3	RESISC45 (Cheng et al., 2017)	45	18,900	6,300
4	EuroSAT (Helber et al., 2019)	10	21,600	2,700
5	SVHN (Netzer et al., 2011)	10	73,257	26,032
6	GTSRB (Stallkamp et al., 2011)	43	26,640	12,630
7	MNIST (LeCun et al., 1998)	10	60,000	10,000
8	DTD (Cimpoi et al., 2014)	47	3,760	1,880
9	Flowers102 (Nilsback & Zisserman, 2008)	102	1,020	6,149
10	PCAM (Veeling et al., 2018)	2	262,144	32,768
11	FER2013 (Goodfellow et al., 2013)	7	28,709	7,178
12	Pets (Parkhi et al., 2012)	37	3,680	3,669
13	STL10 (Coates et al., 2011)	10	5,000	8,000
14	CIFAR100 (Krizhevsky, 2009)	100	50,000	10,000
15	CIFAR10 (Krizhevsky, 2009)	10	50,000	10,000
16	Food101 (Bossard et al., 2014)	101	75,750	25,250
17	Fashion-MNIST (Xiao et al., 2017)	10	60,000	10,000
18	EMNIST Letters (Cohen et al., 2017)	26	124,800	20,800
19	KMNIST (Clanuwat et al., 2018)	10	60,000	10,000
20	Rendered-SST2 (Radford et al., 2021)	2	6,920	1,821
	Total	–	960,174	233,068

Table 7: **Datasets and loaded split sizes.** CLIP8, CLIP14, and CLIP20 contain IDs 1–8, 1–14, and 1–20, respectively.

Subset	k	Dataset IDs from Table 7
8-Std	8	1, 2, 3, 4, 5, 6, 7, 8
8-R1	8	1, 2, 3, 4, 5, 6, 7, 10
8-R2	8	8, 9, 11, 12, 14, 17, 19, 20
8-R3	8	1, 2, 4, 9, 13, 15, 16, 18
8-R4	8	3, 5, 6, 7, 8, 10, 12, 18
14-Std	14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
14-R1	14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16
14-R2	14	1, 2, 3, 4, 5, 6, 8, 13, 14, 15, 17, 18, 19, 20
14-R3	14	1, 5, 7, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20
14-R4	14	2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 18, 20
20-All	20	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

Table 8: **Fixed multi-task evaluation sets.** Dataset IDs refer to Table 7; membership is shared across backbones.

B Implementation and Calibration Details

B.1 Codec Architecture and Initialization

Tensor layout and basis parameterization. Each compressible task-update tensor is reshaped and transposed as needed into $X_{t,\ell} \in \mathbb{R}^{m_\ell \times n_\ell}$ with $m_\ell \geq n_\ell$; its original layout is restored after

decoding. The independent encoder and decoder bases use differentiable thin QR parameterizations to maintain orthonormal columns.

Latent residual maps. Each residual acts independently on columns $z \in \mathbb{R}^{r_\ell}$:

$$R(z) = z + \alpha b \odot \mathrm{GELU}(a \odot \mathrm{RMS}(z)), \qquad \mathrm{RMS}(z) = \frac{z}{\sqrt{\|z\|_2^2/r_\ell + \epsilon}}. \tag{5}$$

The vectors $a, b \in \mathbb{R}^{r_\ell}$ and gate $\alpha \in \mathbb{R}$ are learned separately for each encoder and decoder map; $\odot$ denotes elementwise multiplication and $\epsilon = 10^{-6}$. We use RMS normalization without affine parameters (Zhang & Sennrich, 2019); the activation is GELU (Hendrycks & Gimpel, 2016). Initializing $a = b = \mathbf{1}$ and $\alpha = 0$ makes both maps identities.

Compression and parameter counts. The bases contain $2m_\ell r_\ell$ trainable scalars per compressed tensor, and the residuals contain $4r_\ell + 2$. These parameters are shared across experts and scale linearly with rank for a fixed tensor shape. If the uncompressed scalar/vector path has d_{exact} coordinates, the per-expert code dimension is $d_{\text{code}} = \sum_\ell r_\ell n_\ell + d_{\text{exact}}$. Thus $C = P/d_{\text{code}}$ excludes shared codec storage.

Shared SVD subspace. For $H_\ell = [X_{t,\ell}]_{t \in \mathcal{T}_{\text{train}}}$ with SVD $H_\ell = U_\ell \Sigma_\ell V_\ell^\top$, we take $Q_\ell = U_\ell[:, 1 : r_\ell]$. Equivalently, its columns are the leading eigenvectors of $\sum_t X_{t,\ell} X_{t,\ell}^\top$, avoiding explicit concatenation. At fixed rank this minimizes $\sum_t \|X_{t,\ell} - Q_\ell Q_\ell^\top X_{t,\ell}\|_F^2$. Ranks are allocated using pooled spectral energy per encoded coordinate, with at least one component per compressible tensor, and stay fixed. Initializing $A_\ell = B_\ell = Q_\ell$ as independent copies gives $\widehat{X}_{S,\ell}^{(0)} = Q_\ell Q_\ell^\top \sum_{t \in S} X_{t,\ell}$. All statistics use only the training bank.

B.2 Training and Calibration

MergeCodec. For each backbone, we initialize independent encoder and decoder bases from the training bank's shared SVD subspaces and set nonlinear residual gates to zero. Integer ranks are allocated from pooled spectral energy under the target per-expert code budget $C = 4$. Both the compressed matrix coordinates and the uncompressed path count toward this budget; shared encoder/decoder parameters do not. Consequently, $C = 4$ is a dimension ratio, not a claim of fourfold total storage reduction.

We use training seed 42 and optimize only codec parameters for 4,000 steps with AdamW (Loshchilov & Hutter, 2019), learning rate 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and zero weight decay. The learning rate uses a 10-step warmup and cosine schedule; the global gradient norm is clipped at one. Each selected task contributes a batch of up to 32 training images, including the last partial batch. Cross-entropy is averaged first within each task and then across selected tasks. The pretrained backbone, expert vectors, classification heads, projection, and logit scale remain fixed. CLIP training uses BF16 mixed precision, while codec operations, QR factorization, and parameter synthesis use FP32. Evaluation uses FP32 with TF32 disabled. The final step-4,000 checkpoint is evaluated without selecting a checkpoint by test performance.

The Mixed schedule contains 800 episodes at each of $k \in \{4, 8, 12, 16, 20\}$, randomly interleaved. Balanced construction gives each task exactly 2,400 batch appearances over the full run. This fixed-quota schedule does not imply independent uniform draws of task subsets. Neither singleton episodes nor cardinality 14 appear during training. The full-bank combination is deliberately shared by training and evaluation.

Supervised reference and sampling ablation. MTL(S) fine-tunes an independent image encoder from the common base for each of the 11 selected sets, with fixed task heads. It uses Adam (Kingma & Ba, 2015) at 10^{-5}, a cosine schedule without warmup, and 4,000 updates, each averaging losses from up to 32 images per task in S. It uses FP32 and the fixed final checkpoint. This is a per-set supervised reference; its training data exposure, optimizer, and computational cost are not matched to codec training. The Full20 ablation replaces Mixed sampling with the complete 20-task set in every episode. It has 4,000 updates but 4,000 batch appearances per task, versus 2,400 for Mixed.

Coefficient selection. For each backbone and cardinality, TA selects a coefficient using the standard set's mean test accuracy. At $k = 8$, its grid is $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1\}$; at $k = 14, 20$, it is $\{0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1\}$. TSV-M uses $\{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 2\}$ on ViT-B/32. Its protocol allows a single boundary extension with $\{0.05, 0.1\}$ when the initial optimum is 0 or 0.25, or with $\{3, 4\}$ when it is 2; none of the three standard sets triggers an extension. Ties select the smaller coefficient. The chosen standard-set coefficient is reused on all four random sets with the same cardinality. Table 9 lists the actual coefficients.

Method / backbone	$k = 8$	$k = 14$	$k = 20$
TA / ViT-B/32	0.2	0.1	0.05
TA / ViT-B/16	0.3	0.1	0.05
TA / ViT-L/14	0.2	0.1	0.05
TSV-M / ViT-B/32	1.0	0.75	0.75
MergeCodec / all backbones	1	1	1

Table 9: **Outer merge coefficients.** TA and TSV-M reuse standard-set coefficients across random sets of the same size; Codec fixes $\lambda = 1$.

WUDI and ProbSurgery. WUDI uses 400 optimization iterations with learning rate 10^{-5} and no additional outer rescaling. WUDI+ProbSurgery reuses the resulting WUDI model. Its task-specific correction modules use hidden dimension 128, batch size 16, learning rate 10^{-3}, a KL coefficient of 10^{-4}, and 5,000 adaptation steps per task. Adaptation uses unlabeled test images and expert features; labels are used only for scoring. We evaluate the fixed final step and retain the correction modules at inference. This adaptation is performed separately for each merged combination.

Matched ablation controls. Structure variants use the same SVD initialization, ranks, Mixed episode order, and image batches, with both bases trainable. Mixed and Curriculum share the same episodes and batches, differing only in order. The initialization ablation uses identical Mixed episodes, image batches, and optimizer settings. Random initialization orthonormalizes Gaussian matrices, initializing both bases from the same subspace as independent trainable copies; the SVD-derived rank allocation is retained.

B.3 Compression Factor and Storage

The $C \in \{2, 4, 8\}$ comparison uses the two-sided nonlinear codec under Mixed training, with the same episode order, selected tasks, image batches, optimizer settings, and 4,000-update endpoint. Each setting uses its corresponding SVD initialization and rank allocation. The $C = 4$ reference is the same checkpoint used in the main experiments. Evaluation fixes $\lambda = 1$ and uses the same 20 singleton and 11 multi-task combinations. The $k = 8, 14$ scores average five combinations, each of which averages tasks equally.

Table 10 separates shared codec parameters from cached codes. C is the target ratio of original task-vector dimension to per-expert code dimension, including the uncompressed coordinates; integer rank allocation gives actual ratios of 2.025, 4.000, and 8.000. It is not a total-system storage compression ratio. Both trained bases and nonlinear maps are counted in the shared payload. Increasing C from four to eight reduces the shared-plus-20-code payload to 50.59% of its former size. The corresponding mean accuracy losses are 0.44 points for singleton recovery and 0.31-0.40 points for multi-task merging. Reducing C to two improves all 11 multi-task set means, but uses 1.88 times the shared-plus-code payload of $C = 4$. These results quantify the observed capacity-accuracy tradeoff.

Target C	Actual C	Params (M)	Shared (MiB)	Code/task (MiB)	Shared + 20 codes (GiB)
2	2.025	174.89	667.15	164.73	3.869
4	4.000	114.75	437.75	83.40	2.056
8	8.000	60.63	231.27	41.70	1.040

Table 10: **Compression and storage on ViT-B/32.** FP32 tensor payloads; Base, heads, optimizer states, and file metadata excluded.

C Complete Subset and Reconstruction Results

Table 11 includes all ViT-B/32 combinations, including the five individual 14-task sets summarized in the main text. Table 12 provides the corresponding results for ViT-B/16 and ViT-L/14. Scores are unweighted means across the tasks in each row. The Full20 and Mixed columns use the same fixed outer coefficient, $\lambda = 1$. Full20 can remain competitive on its training combination: on ViT-B/32 with all 20 tasks, it obtains 88.81%, versus 88.73% for Mixed.

Subset	Base	Experts	MTL(S)	TA	TSV-M	WUDI	WUDI+PS	Full20	Codec
8-Std	48.10	90.38	89.60	69.51	83.07	84.77	88.65	85.85	**89.31**$_{+0.66}$
8-R1	50.15	91.42	90.98	70.77	84.47	86.52	89.49	87.65	**90.82**$_{+1.33}$
8-R2	53.85	85.61	85.14	63.79	78.37	80.43	**84.01**	79.28	83.97$_{-0.04}$
8-R3	64.45	90.02	90.09	71.04	82.70	84.77	88.63	84.02	**89.14**$_{+0.51}$
8-R4	46.61	93.32	92.72	67.15	80.64	83.88	91.82	88.49	**92.43**$_{+0.61}$
14-Std	56.97	89.25	88.51	66.49	79.05	78.82	86.23	86.33	**88.00**$_{+1.77}$
14-R1	57.78	89.26	88.66	66.94	79.74	79.34	86.40	85.83	**88.11**$_{+1.71}$
14-R2	52.20	90.49	89.85	58.87	73.01	73.82	87.04	88.41	**89.43**$_{+2.39}$
14-R3	55.83	89.03	88.67	63.37	74.19	74.63	85.77	85.59	**88.03**$_{+2.26}$
14-R4	57.11	89.54	88.92	64.73	77.58	78.78	87.79	86.52	**88.73**$_{+0.94}$
20-All	55.66	89.77	89.07	61.10	73.56	66.57	84.41	88.81	**88.73**$_{+4.32}$

Table 11: **Complete ViT-B/32 subset results (accuracy, %).** Full20: full-bank sampling ablation. Bold: best merging method. Subscripts: Codec minus the best of TA, TSV-M, WUDI, and WUDI+PS (pp; green/red: gain/loss). Full20 and reference models are excluded from ranking and differences.

Subset	Base	Experts	MTL(S)	TA	Full20	Codec	Δ_{MTL}
ViT-B/16							
8-Std	55.36	92.37	92.00	77.13	85.11	91.16	-0.84
8-R1	56.48	93.41	93.09	78.20	87.84	92.55	-0.54
8-R2	57.07	87.70	88.06	65.80	84.06	87.15	-0.91
8-R3	68.01	92.79	92.82	70.92	88.13	92.08	-0.74
8-R4	51.70	94.43	94.16	70.63	89.67	93.56	-0.60
14-Std	61.97	91.33	90.96	70.93	88.61	90.47	-0.49
14-R1	62.92	91.56	91.16	71.72	88.62	90.70	-0.46
14-R2	57.00	92.01	91.88	62.70	90.04	91.08	-0.80
14-R3	59.47	90.82	90.95	66.36	89.52	90.31	-0.63
14-R4	60.54	91.61	91.55	68.03	89.16	91.21	-0.35
20-All	59.85	91.62	91.35	64.81	91.25	91.08	-0.27
ViT-L/14							
8-Std	64.94	94.31	94.04	82.07	87.41	93.63	-0.41
8-R1	64.40	95.02	94.82	82.77	90.07	94.58	-0.24
8-R2	62.34	90.23	90.87	71.11	87.34	89.70	-1.17
8-R3	73.13	95.11	95.19	80.27	85.43	94.85	-0.33
8-R4	58.62	95.26	95.12	76.06	86.78	94.98	-0.13
14-Std	69.12	93.37	93.13	77.87	91.71	92.81	-0.32
14-R1	70.00	93.49	93.35	78.35	92.18	93.05	-0.30
14-R2	62.22	93.97	93.98	68.98	93.03	93.37	-0.61
14-R3	64.31	92.67	93.00	70.92	92.08	92.25	-0.76
14-R4	67.34	93.47	93.75	74.46	92.33	93.10	-0.65
20-All	65.62	93.53	93.60	71.14	93.21	92.98	-0.62

Table 12: **Complete ViT-B/16 and ViT-L/14 subset results (accuracy, %).** Full20: full-bank sampling ablation; Codec: Mixed training. Δ_{MTL}: Codec minus MTL(S), in percentage points. TA coefficients appear in Table 9.

D Single-Expert Reconstruction Scores

Table 13 supplies the 60 expert/reconstruction pairs plotted in Figure 4. Each reconstruction uses only that expert's encoded task vector and the same frozen codec used for multi-task composition. This experiment changes the input cardinality to one without additional training or coefficient tuning. Small mean accuracy losses coexist with both gains and losses on individual tasks.

Dataset	B/32 Expert	B/32 Codec	B/16 Expert	B/16 Codec	L/14 Expert	L/14 Codec
SUN397	74.84	75.15$_{+0.31}$	78.92	78.74$_{-0.18}$	82.76	81.99$_{-0.77}$
Cars	78.56	78.78$_{+0.22}$	85.88	86.17$_{+0.29}$	92.77	92.05$_{-0.72}$
RESISC45	95.14	94.71$_{-0.43}$	96.57	95.59$_{-0.98}$	97.40	97.13$_{-0.27}$
EuroSAT	99.07	98.78$_{-0.30}$	99.04	98.70$_{-0.33}$	99.15	99.19$_{+0.04}$
SVHN	97.27	96.32$_{-0.94}$	97.61	96.87$_{-0.74}$	97.92	97.48$_{-0.45}$
GTSRB	98.91	98.57$_{-0.33}$	98.99	98.57$_{-0.42}$	99.24	99.30$_{+0.06}$
MNIST	99.58	99.59$_{+0.01}$	99.70	99.63$_{-0.07}$	99.76	99.71$_{-0.05}$
DTD	79.68	78.99$_{-0.69}$	82.23	82.66$_{+0.43}$	85.48	84.79$_{-0.69}$
Flowers102	88.55	90.34$_{+1.79}$	94.88	95.58$_{+0.70}$	97.67	96.81$_{-0.86}$
PCAM	87.96	90.17$_{+2.21}$	90.55	90.98$_{+0.43}$	91.14	91.83$_{+0.69}$
FER2013	71.58	70.13$_{-1.45}$	72.82	72.62$_{-0.20}$	75.93	75.48$_{-0.45}$
Pets	92.42	92.48$_{+0.05}$	94.49	94.74$_{+0.25}$	95.75	95.94$_{+0.19}$
STL10	97.55	98.30$_{+0.75}$	98.16	98.89$_{+0.72}$	99.22	99.42$_{+0.20}$
CIFAR100	88.37	87.24$_{-1.13}$	88.77	87.95$_{-0.82}$	93.02	92.21$_{-0.81}$
CIFAR10	97.60	97.45$_{-0.15}$	98.27	98.13$_{-0.14}$	99.13	99.16$_{+0.03}$
Food101	88.41	87.75$_{-0.67}$	91.88	91.51$_{-0.37}$	94.76	94.57$_{-0.20}$
Fashion-MNIST	94.75	94.03$_{-0.72}$	94.54	94.31$_{-0.23}$	95.28	95.03$_{-0.25}$
EMNIST Letters	95.61	94.55$_{-1.06}$	95.28	94.78$_{-0.50}$	95.43	95.15$_{-0.28}$
KMNIST	98.23	97.83$_{-0.40}$	98.10	97.21$_{-0.89}$	98.30	97.82$_{-0.48}$
Rendered-SST2	71.28	69.08$_{-2.20}$	75.73	74.52$_{-1.21}$	80.45	82.65$_{+2.20}$
Mean	89.77	89.51$_{-0.26}$	91.62	91.41$_{-0.21}$	93.53	93.38$_{-0.14}$

Table 13: **Single-expert reconstruction accuracy (%).** Expert: original model; Codec: reconstruction by the frozen Mixed codec at $\lambda = 1$. Gray subscripts: Codec minus Expert, in percentage points.

E Unseen-Expert Protocol and Additional Results

Expert partition and training. We use a fixed expert partition generated with split seed 20260917. Table 14 defines the nested training banks. SVHN, Flowers102, PCAM, and Rendered-SST2 are excluded from every bank. Their task vectors, images, and labels are not used to fit the SVD bases, allocate layer ranks, or optimize the codec. Each bank starts from its own SVD initialization, without inheriting a smaller-bank checkpoint or the 20-expert initialization. The common base, visual projection, and logit scale remain frozen, as do the text-derived classification heads. Only training-task heads are used during optimization. Held-out expert weights become available when constructing the evaluation models.

The optimizer and precision settings follow Appendix B. Each run uses 4,000 updates and the five cardinality slots in Table 14, with 800 episodes per slot. The size-two slot occurs twice when $n = 4$. Every training task contributes exactly 2,400 batches, with matching image-index streams for tasks shared across banks. This controls per-task exposure and update count, but not total training data or computation. SVD rank allocation is repeated within the common $C = 4$ dimension budget, so the shared codec parameter count also varies. The trained codecs are evaluated at their final checkpoints with $\lambda = 1$, full test splits, FP32, and no target-task adaptation.

n	New dataset IDs	Episode cardinalities	Task batches	Codec params (M)
4	7, 14, 15, 19	1, 2, 2, 3, 4	9,600	113.18
8	2, 4, 11, 16	2, 3, 5, 6, 8	19,200	114.86
12	12, 13, 17, 18	2, 5, 7, 10, 12	28,800	114.52
16	1, 3, 6, 8	3, 6, 10, 13, 16	38,400	115.20

Table 14: **Nested OOD training banks.** n: training experts; dataset IDs refer to Table 7. Task batches are totals across the bank; M denotes millions.

Baseline calibration and references. TA and TSV-M each select one coefficient per k on the complete official test splits of the four target tasks. Selection maximizes the same task-balanced score reported on the fixed panel: six pairs for $k = 2$ and the four-task set for $k = 4$. Thus these are target-test-selected references (best within each candidate grid), using the same test labels for selection and reporting. One coefficient is shared across all combinations of a given size and all n. The unseen-expert condition refers to exclusion from Codec initialization and training; baseline coefficient selection accesses the target test labels.

TA uses $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1\}$ and allows one boundary extension: $\{0.025, 0.05\}$ if the initial best is 0 or 0.1, or $\{1.5, 2, 3\}$ if it is 1. TSV-M uses the eight-point grid and extension rule in Appendix B. Neither method requires an extension for these runs; ties favor the smaller coefficient. The two methods have different grids. Table 15 lists the selected coefficients. WUDI uses 400 iterations at learning rate 10^{-5}, without outer rescaling or target-image adaptation. Its inputs do not depend on n, so its seven held-out merges are reused across all four rows. TA and TSV-M also reuse their selected models and scores across n, since the target panel and selected coefficients are fixed. Repeated baseline values are not independent experimental replicates. We verify all 256 task-coefficient evaluations and recompute the four2 selection curves from correct/total counts; 168 evaluations are new and 88 reuse identity-checked results.

k	TA	TSV-M
2	0.5	0.75
4	0.4	0.75

Table 15: **Target-test-selected OOD coefficients.** Each coefficient is shared across all combinations of size k and all n.

Frozen SVD-T retains each run's untrained initialization, including its bases, ranks, uncompressed path, and fixed $\lambda = 1$. Codec-20 is the main Mixed codec trained on all 20 experts, including the four held-out tasks. It is an all-seen reference, distinct from the Full20 sampling ablation.

Fixed held-out panel. Table 16 reports all eight multi-task comparisons. The base and separate-expert means on this panel are 54.31% and 86.27%, respectively. For each method we first average a task's accuracies over the combinations containing it, then average across the four tasks. Each task appears in three pairs and in the single four-task set, so this balanced panel also equals the unweighted average of subset scores. The same four tasks are used at every n. Additional panels over all remaining unseen experts have changing membership and are not used to infer a bank-size trend.

k	n	TA†	TSV-M†	WUDI	Codec	Frozen SVD-T	Codec-20
2	4	81.82	83.44	**84.73**	82.81	72.54	86.34
2	8	81.82	83.44	**84.73**	84.13	72.38	86.34
2	12	81.82	83.44	**84.73**	83.98	72.10	86.34
2	16	81.82	83.44	**84.73**	84.64	72.19	86.34
4	4	77.71	**81.62**	81.52	79.01	58.75	86.00
4	8	77.71	81.62	81.52	**82.42**	58.62	86.00
4	12	77.71	81.62	81.52	**82.16**	58.50	86.00
4	16	77.71	81.62	81.52	**83.15**	58.73	86.00

Table 16: **Fixed-panel OOD accuracy on ViT-B/32 (%).** Bold: best of TA, TSV-M, WUDI, and Codec. †: target-test-selected coefficients. Frozen SVD-T: untrained initialization; Codec-20: all-seen reference.

TA selects $\lambda = 0.5$ for pairs and 0.4 for four-task merging, yielding 81.82% and 77.71%, respectively. TSV-M selects $\lambda = 0.75$ for both sizes, yielding 83.44% and 81.62%. MergeCodec exceeds these TA references in all eight comparisons and TSV-M in six; both $n = 4$ settings remain below TSV-M. Its advantage over WUDI is confined to the three four-task settings with $n \geq 8$.

Task-level transfer and reconstruction. Table 17 shows that MergeCodec can combine unseen experts to substantially improve on the pretrained Base model, even though all four target tasks are excluded from codec initialization and training. At $n = 16$, merging their task vectors improves accuracy over Base on all four tasks, with gains ranging from 12.33 to 62.81 percentage points. Some tasks also benefit from a larger training bank: increasing n from four to 16 improves Flowers102 by 12.25 points and PCAM by 4.35 points. Moreover, at $n = 16$, MergeCodec outperforms WUDI on three of the four unseen tasks—Flowers102, PCAM, and Rendered-SST2—by 5.58, 2.64, and 1.05 points, respectively. These results support the transfer of the learned composition ability to experts whose tasks were not encountered during codec training.

For singleton inputs from the same four held-out experts, mean reconstruction accuracy is 84.08%, 84.96%, 85.11%, and 85.71% for $n = 4, 8, 12, 16$, respectively. The original experts average 86.27%, so the final reconstruction gap is 0.56 points. Frozen SVD-T obtains 83.63% at $n = 16$.

Task	Base	Expert	Codec-4	Codec-8	Codec-12	Codec-16	TA†	TSV-M†	WUDI
SVHN	31.59	97.27	94.65	94.39	94.11	94.40	90.10	94.64	97.11
Flowers102	66.43	88.55	66.51	77.05	76.68	78.76	66.32	75.88	73.18
PCAM	60.62	87.96	82.72	86.51	87.57	87.07	82.92	82.77	84.43
Rendered-SST2	58.59	71.28	72.16	71.72	70.29	72.38	71.50	73.20	71.33

Table 17: **Four-expert OOD accuracy by task (%).** Base: pretrained CLIP; Expert: each task's original expert. Both references are fixed across training-bank sizes. Codec suffixes denote training-bank size; †: target-test selection.

These are functional recovery scores; training optimizes classification loss rather than exact parameter reconstruction.

Seen/unseen mixtures. At $n = 16$ we also hold the merged cardinality at four while varying the number r of unseen experts. Each combination contains $4 - r$ seen experts from the training bank and r unseen experts from the held-out panel. We evaluate ten combinations for each $r = 0, 1, 2, 3$ and the single all-unseen combination at $r = 4$. For each combination, we average accuracy equally over all four participating tasks, including both seen and unseen tasks; each row then averages these scores equally over its combinations. Base, calibrated TA, and Codec are evaluated on the same tasks and combinations. TA uses the fixed calibrated coefficient $\lambda = 0.4$ for every combination, while Codec uses $\lambda = 1$ without further calibration. Table 18 reports these overall scores.

r	Seen + unseen	Sets	Base	Calibrated TA $\lambda = 0.4$	Codec	Codec–TA
0	$4 + 0$	10	57.19	78.84	**89.75**	+10.92
1	$3 + 1$	10	54.26	76.81	**86.80**	+9.99
2	$2 + 2$	10	54.49	77.75	**85.66**	+7.91
3	$1 + 3$	10	56.06	78.12	**84.87**	+6.76
4	$0 + 4$	1	54.31	77.71	**83.15**	+5.44

Table 18: **Seen/unseen mixtures at $n = 16$, $k = 4$ (accuracy, %).** r: unseen experts per combination. Each score averages all four tasks within each combination, then averages over combinations. Codec–TA: gain in percentage points, computed before rounding. Bold: best method per row.

Codec outperforms both Base and calibrated TA at every mixing ratio, with gains of 5.44–10.92 percentage points over TA. Accuracy remains high from all-seen inputs (89.75%) to all-unseen inputs (83.15%), supporting reuse of the same frozen codec across combinations of familiar and unseen experts.

F Efficiency and Transfer Beyond the Merge Inputs

F.1 Training Efficiency

We profile Codec Mixed training and independent MTL8, MTL14, and MTL20 runs on the same H100 80GB. Each uses 10 warmup and 100 measured updates, real training images, per-task batch/microbatch size 32, BF16 visual forward, and FP32 parameters and optimizer states, with TF32 disabled. Codec composition and vector-Jacobian products use FP32. Each Mixed cardinality $k \in \{4, 8, 12, 16, 20\}$ contributes 20 measured updates. The original method-specific optimizers, loss normalization, and clipping are retained, so this comparison measures training pipelines rather than isolating architectural overhead.

Training time starts after CPU batches are ready and includes transfer, forward/backward passes, composition, optimization, and CUDA synchronization. Loop time additionally includes batch waiting; neither includes setup, logging, evaluation, or checkpoint saving. Existing SVD initialization and experts are reused. These are single short profiles, not completed 4,000-step runs or estimates of time to equal accuracy. In particular, the BF16 MTL timings must not be paired with the main table's historical FP32 MTL accuracy as a measured accuracy-cost tradeoff.

Codec has greater per-update cost and peak memory than MTL20 here. Its reuse can amortize training over independently requested MTL(S) models.

Backbone	Method	Mean k	Train (s)	Loop (s)	Images/s	GiB
ViT-B/32	Codec	12	0.989	0.990	388.3	19.43
ViT-B/32	MTL8	8	0.157	0.618	1630.8	2.28
ViT-B/32	MTL14	14	0.300	0.617	1492.7	2.28
ViT-B/32	MTL20	20	0.410	0.620	1560.0	2.28
ViT-B/16	Codec	12	1.023	1.024	375.3	21.15
ViT-B/16	MTL8	8	0.175	0.624	1465.1	4.28
ViT-B/16	MTL14	14	0.317	0.612	1411.2	4.28
ViT-B/16	MTL20	20	0.479	0.592	1337.1	4.28
ViT-L/14	Codec	12	2.565	2.566	149.7	73.43
ViT-L/14	MTL8	8	0.549	0.653	466.7	14.44
ViT-L/14	MTL14	14	0.974	0.991	460.0	14.44
ViT-L/14	MTL20	20	1.378	1.380	464.3	14.44

Table 19: **Controlled training profiles.** Mean seconds per update; loop includes batch waiting. Memory is peak GPU allocated GiB.

F.2 Merging Efficiency

We time the standard eight- and fourteen-task sets and the complete 20-task set on ViT-B/32, using FP32 and disabling TF32. Inputs already reside on the GPU; each merge generates a fresh complete parameter mapping, including addition to the base. Disk loading, saving, coefficient search, and accuracy evaluation are excluded. TA and TSV-M reuse the coefficients from the main comparison. TA and both Codec paths use five warmups and 30 repetitions; TSV-M and WUDI use one warmup and three repetitions. We report synchronized wall-time medians.

The Codec encode path includes encoding, aggregation, and decoding; the cached path retains codes, decoder, and base, releasing the encoder and raw vectors. Both exclude the separately measured QR export of trained bases. Cached codes additionally require one-time encoding. These are post-training costs; SVD fitting and codec training are not included. TSV-M uses the existing full decomposition implementation without an additional per-expert SVD cache, so cached-Codec ratios are not claims about every optimized implementation.

Stage	$k = 8$	$k = 14$	$k = 20$
Codec QR (cached path)	0.355	0.350	0.352
Codec QR (encode path)	0.382	0.343	0.360
Codec encoding	0.069	0.117	0.176
PS expert features (median)	144.153	199.186	245.072

Table 20: **One-time preparation (seconds).** Codec: single observations; PS: medians. QR exports trained bases, not SVD initialization.

WUDI uses 400 optimization iterations. WUDI+PS adds model installation, merged-model feature extraction, and 5,000 adaptation steps per task, using cached expert features. Its total is computed by summing stages within each of three repetitions and then taking the median. Feature extraction includes image reading/preprocessing, transfers, forward passes, and CPU feature materialization. The method retains task-specific adapters.

F.3 Tasks Omitted from the Merge Inputs

We reuse the main ViT-B/32 Mixed codec trained on all 20 tasks at $C = 4$, but supply only the standard CLIP8 task vectors at evaluation. We score the remaining 12 tasks on all 145,635 test images, averaging task accuracies equally. These tasks participated in codec training and SVD initialization, so this is not an unseen-expert OOD experiment. The fixed task heads and preprocessing match the Base evaluation. TA's $\lambda = 0.2$ was selected on CLIP8, without using the other 12 tasks. Frozen SVD-20 shares the codec's initial subspaces and ranks but receives no supervised codec training. Every merged model uses only the same eight input experts; Codec and TA model hashes match their CLIP8 evaluations.

Figure 6 uses MTL8 trained on CLIP8 as the supervised reference for the eight input tasks. For the other 12 tasks, it reports the corresponding task-balanced mean from MTL20 trained on all 20 tasks. These are two different reference models; MTL20 has received supervision on the omitted

tasks and is not an MTL8 transfer baseline. The archive contains no evaluation of MTL8 on these 12 tasks. We verify the MTL test counts against the transfer evaluation and average the same per-task accuracies used in the main MTL comparison.

Task	Base	TA, $\lambda = 1$	TA, $\lambda = 0.2$	Frozen SVD-20	Codec	MTL20
Flowers102	66.43	3.81	60.19	4.03	71.65	88.27
PCAM	60.62	50.02	54.06	50.02	51.31	87.39
FER2013	41.49	10.24	40.37	11.38	46.87	71.37
Oxford-IIIT Pet	83.32	6.57	81.17	6.65	81.25	91.77
STL10	97.12	16.81	94.03	17.06	93.71	98.26
CIFAR100	63.71	4.60	55.45	4.66	63.40	86.85
CIFAR10	89.82	15.31	84.50	15.40	84.83	97.57
Food101	82.36	2.65	71.57	2.65	76.86	88.27
Fashion-MNIST	63.02	25.06	64.44	25.45	65.24	93.75
EMNIST Letters	12.00	7.36	14.64	6.88	18.79	94.83
KMNIST	9.95	8.79	9.14	8.57	12.41	96.51
Rendered-SST2	58.59	49.92	54.64	49.92	52.55	72.32
Mean	60.70	16.76	57.02	16.89	59.91	88.93

Table 21: **Tasks omitted from the merge inputs (accuracy, %).** Codec and TA use only CLIP8 experts. MTL20 is supervised on all 20 tasks; TA's 0.2 is selected on CLIP8.

On the 12 tasks omitted from the merge inputs, Codec achieves 59.91% mean accuracy, comparable to Base (60.70%) and calibrated TA (57.02%): it is 0.79 points below Base and 2.89 points above TA. Thus, although these tasks participated in codec training, their knowledge is not automatically expressed as improved average performance when their task vectors are absent. In contrast, Codec reaches 89.31% on the eight supplied tasks, close to the corresponding MTL8 reference (89.60%). This pattern supports the interpretation that high merging performance comes from composing the supplied task vectors, rather than broadly injecting task knowledge acquired during codec training into the merged model. Together with the strong merging results on experts excluded from codec initialization and training (Section 4.3), this provides further evidence that MergeCodec learns a reusable ability to compose the task vectors it receives.