LS-Merge: Merging Language Models in Latent Space

Bedionita Soro [*1] **Aoxuan Silvia Zhang** [*1] **Bruno Andreis** [2] **Jaehyeong Jo** [1] **Song Chong** [1] **Sung Ju Hwang** [3 1]

Abstract

Model merging in weight space efficiently reuses pretrained models, but existing methods assume matching architectures, making heterogeneous merges brittle or infeasible. We address this by encoding weights into a smooth latent space where cross-architecture merging becomes natural, then decoding the merged representation back to weights. Two challenges arise: the billions of parameters in LLMs make latent encoding costly, and high compression ratios can hinder generalization to unseen weights. We tackle both with a transformer-based VAE trained via a two-stage compression curriculum with layer-aware chunking, first learning a high-capacity representation and then distilling it to a compact code for stability and out-of-distribution generalization. A dimensionality-matching projection and distributional alignment via optimal transport further enables interpolation across different model sizes. Empirically, latent-space interpolation is more robust than direct weight averaging and yields stronger downstream performance across heterogeneous merges. Together, these components provide a scalable recipe for model merging across heterogeneous architectures.

1. Introduction

Pretraining large language models (LLMs) demands millions of GPU-hours, making efficient reuse of the resulting weights a pressing concern. Weight-space model merging addresses this by combining parameters from multiple pretrained models to integrate complementary capabilities without additional training (Yang et al., 2024). Existing methods, from linear interpolation (Wortsman et al., 2022) and task arithmetic (Ilharco et al., 2023) to evolutionary search (Akiba et al., 2025), have demonstrated consistent practical gains at scale. However, they share two fundamental limitations: they require *architecturally homogeneous* models with matching shapes, and they typically need at least two distinct checkpoints, precluding single-model augmentation (Yu et al., 2024a).

Recent work on weight-space learning (Sch"urholt et al., 2024; Peebles et al., 2022; Knyazev et al., 2025) shows that neural network parameters can be projected into smooth, fixed-dimensional latent spaces using generative models such as VAEs (Kingma & Welling, 2013) or diffusion models (Wang et al., 2025). While successful in vision, applying these ideas to LLM-scale weights, and exploiting them for merging, remains largely unexplored. We propose **LS-Merge**, a framework that encodes LLM weights into a learned latent space, performs merging operations there, and decodes back to parameters (Figure 1). By operating in latent space, LS-Merge supports both homogeneous merging (including self-merging from a single checkpoint) and heterogeneous merging across different depths, widths, or model families. Our key contributions are:

- **Weight statistics that matter for encoding.** We show that LLM weights exhibit low variance with heavy tails, motivating encoder designs that preserve rare high-magnitude parameters rather than over-regularizing toward a Gaussian.
- **Merging LLMs in latent space.** We propose LS-Merge, a latent-space merging methodology that encodes LLM weights via a transformer-based VAE trained with a two-stage curriculum and structured layer-aware chunking, and merges them by interpolating in the learned latent space.
- **Heterogeneous merging.** We introduce a dimensionality-matching projection and OT-based latent alignment that enables interpolation between models of different depths or widths.
- **Consistent empirical gains.** On Gemma and LLaMA pretrained models and LoRA-expert merging, our

[1]KAIST AI [2]Department of Engineering, Oxford University [3]Deepauto.ai. Correspondence to: Bedionita Soro <sorobedio@kaist.ac.kr>.

Workshop on Weight-Space Symmetries, held in conjunction with the 43rd International Conference on Machine Learning, Seoul, South Korea. 2026. Copyright 2026 by the author(s).

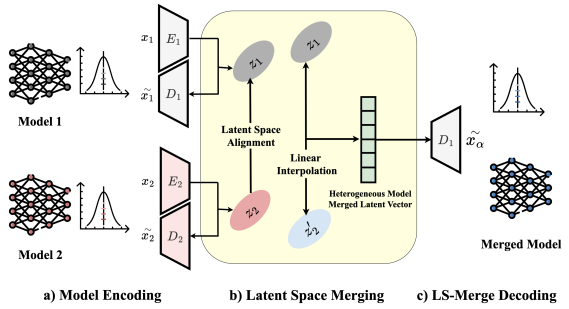


Figure 1. **LS-Merge: encode, align, and decode for LLM weight merging.** **(a)** Pretrained weight tensors x_1 and x_2 are chunked layer-wise and encoded by E_1, E_2 into latents z_1, z_2; decoders D_1, D_2 reconstruct $\tilde{x}_1, \tilde{x}_2$. **(b)** A latent-space alignment places z_1 and z_2 in a shared manifold where they are merged(linear interpolation) yielding a merged latent z_α, supporting both homogeneous (same-architecture(same encoding model) and heterogeneous (cross-architecture) merges. **(c)** A target decoder (e.g., D_1) maps z_α back to weights $\tilde{x}_\alpha$, producing the merged model. Here, x_i denotes the full weight set of model i, z_i its latent code, and tildes indicate reconstructions.

method outperforms existing merging methods and remains robust under heterogeneity.

2. Related Work

Weight-space merging. Model merging directly combines parameters from multiple checkpoints. Simple approaches include weight averaging, Model Soup (Wortsman et al., 2022), and spherical interpolation (SLERP). Task Arithmetic (Ilharco et al., 2023) represents task-specific behavior as parameter-space vectors, while TIES-Merging (Yadav et al., 2023), DARE (Yu et al., 2024a), and related methods (Yu et al., 2024b) address interference through sparsification, rescaling, or sign alignment. Greedy Soup (Wortsman et al., 2022) and EvolMerge (Akiba et al., 2025) further treat merging as a search problem over candidate recipes. These methods are effective but generally require shape-compatible models and operate directly in weight or delta space.

Modular composition. Another line of work combines modules or experts rather than merging full checkpoints. Model Stocks (Jang et al., 2024), LoraHub (Huang et al., 2023), Pack of LLMs (Mavromatis et al., 2024), and cBTM (Gururangan et al., 2023) compose specialized models or adapters to improve task performance. While flexible, these methods often increase inference cost or do not yield a single unified checkpoint. In contrast, LS-Merge performs merging in a learned latent space and decodes the result into one parameter set, relaxing the need for direct architectural alignment.

3. LS Merge

We present **LS-Merge**, a framework for model merging *in latent space* (Figure 1): pretrained LLM weights are encoded into a learned latent space, merged there, with alignment when models are heterogeneous, and decoded back to parameter space.

3.1. Weight Preprocessing and Encoding

We standardize pretrained weights of varying shapes offline, then learn a compact latent representation with a transformer-based VAE, chosen for its generalization to unseen checkpoints and training efficiency over convolutional alternatives.

Preprocessing as a Sequence. Each layer's weight tensor is flattened to $w \in \mathbb{R}^L$, zero-padded to $L_p = \lceil L/c \rceil c$, and split into $n = L_p/c$ chunks of size c, forming $X \in \mathbb{R}^{B\times n\times c}$. Chunks are linearly embedded to $X_{\text{emb}} \in \mathbb{R}^{B\times n\times d}$ and processed by a transformer encoder E_θ with optional token downsampling:

$$z = E_\theta(X_{\text{emb}}) \in \begin{cases} \mathbb{R}^{B\times z_d}, & \text{pooled}, \\ \mathbb{R}^{B\times n\times z_d}, & \text{token-wise}. \end{cases} \tag{1}$$

A decoder D_ϕ reconstructs weights from z via the β-VAE objective:

$$\mathcal{L} = -\mathbb{E}_{q_\phi(z\,|\,w)}\big[\log p_\theta(w\,|\,z)\big] + \beta\,\mathrm{KL}(q_\phi(z\,|\,w)\,\|\,p(z))\,, \tag{2}$$

with $p(z) = \mathcal{N}(0, I)$ and fixed β.

3.2. Latent Space Alignment and Merging

Self-Merging and Homogeneous Merging *Self-merging* encodes a single model and draws multiple latent codes from its posterior (or the prior) to combine them, which is equivalent to merging *homogeneous* models whose per-layer embeddings share the same dimensionality. To be specific, for two checkpoints with weights $W_a, W_b \in \mathbb{R}^N$, we encode $z_a = E(W_a)$ and $z_b = E(W_b)$, and linearly interpolate them as $z_\lambda = (1-\lambda)z_a + \lambda z_b$ for $\lambda \in [0, 1]$. We obtain the weights by decoding the interpolated latent $\widehat{W}_\lambda = D(z_\lambda)$. Empirically, latent-domain interpolation better preserves functional coherence than direct weight-space mixing, and common merge operators, for example, model soup or task arithmetic (Wortsman et al., 2022; Ilharco et al., 2023), transfer naturally by applying them to $\{z_a, z_b\}$ before decoding.

Heterogeneous Mapping. When two architectures match layer by layer in chunk count, a single VAE embeds all layers into a shared d-dimensional latent space. When their depths, widths, or per-layer chunk counts differ, we use

separate encoders for each architecture and map their per-layer latents to a common dimension.

Let the source have n_s layers of size M and the target have n_t layers of size N. We project the aligned source and target latents to

$$Z^{\text{src}} \in \mathbb{R}^{n_t \times d}, \qquad Z^{\text{tgt}} \in \mathbb{R}^{n_t \times d}. \tag{3}$$

This standardizes the latent shape across architectures and enables interpolation with standard merge operators. However, when VAEs are trained separately, equal shape does not guarantee that the latent distributions are geometrically comparable.

Optimal Transport Alignment. Homogeneous models, such as checkpoints from the same pre-training run, often exhibit overlapping latent support and admit well-behaved linear interpolation. In contrast, heterogeneous models such as Gemma and Llama can produce latent distributions on distinct manifolds with different covariance structures. Direct interpolation between such misaligned latents may move outside the target decoder's well-trained latent region and degrade downstream performance.

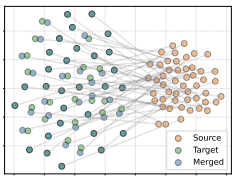


Figure 2. Visualization of embeddings of Gemma models and OT-aligned latents. The merged latent partially overlaps with the target latent.

We therefore treat heterogeneous merging as a manifold registration problem and align the source latent distribution to the target distribution using Optimal Transport (OT) (Villani, 2009; Santambrogio, 2015). Specifically, we estimate a transport map

$$T^* : \mathcal{Z}_{\text{src}} \to \mathcal{Z}_{\text{tgt}} \tag{4}$$

that moves source latents toward the target latent distribution while minimizing geometric distortion. The full OT formulation is given in Appendix B. Under a Gaussian approximation of each layer's latent distribution, this map has a closed-form affine solution that aligns the source mean and covariance to the target. In practice, we use the POT library (Flamary et al., 2021; 2024).

After alignment, we interpolate in the transported latent space:

$$Z_\lambda^{\text{OT}} = (1-\lambda) Z_{\text{tgt}} + \lambda \tilde{Z}_{\text{src}}. \tag{5}$$

The aligned latent Z_λ^{OT} is then decoded by the target decoder to obtain the merged model. The full procedure is summarized in Algorithm 1.

Extension to N-Model Merging. We extend pairwise merging to N models, such as LoRA experts, using convex barycentric interpolation. Given latent codes $z_i \sim q(z \mid W_i)$ for each expert W_i, we merge them as

$$z_{\text{merged}} = \sum_{i=1}^{N} \lambda_i z_i, \qquad \lambda_i \geq 0, \qquad \sum_{i=1}^{N} \lambda_i = 1. \tag{6}$$

The merged weights are decoded as

$$\widehat{W}_{\text{merged}} = D(z_{\text{merged}}). \tag{7}$$

Uniform averaging and Greedy Soup correspond to different choices of the weights $\{\lambda_i\}$.

4. Experiments

Setup. We evaluate latent-space merging on Gemma and LLaMA models, including `Gemma-3-1B-it`, `Gemma-3-4B-it`, `Llama-3-1B-instruct`, and `Llama-2-7b`. For expert fusion, we benchmark 10 LoRA experts on `Gemma-7B-it`. Following Feng et al. (2024b), we evaluate across language understanding, commonsense reasoning, math, and knowledge-intensive tasks. We compare against reference-free weight-space methods: SLERP, Uniform Soup and Greedy Soup (Wortsman et al., 2022), data-level merging, raw pretrained checkpoints, and the best single expert. Full dataset and baseline details are provided in Appendix C.

Evaluation Protocols We evaluate compression and merging in four scenarios:

1. **Self-Merging:** sample multiple codes from one model's latent distribution and merge them.
2. **Expert Merging:** merge LoRA experts in latent space vs. weight space.
3. **Cross-Architecture Merging:** align and merge models with various depths and widths.
4. **Ablation study:** reconstruction or compression behavior and generalization to unseen checkpoints.

Weight-Encoding Model. We train a Transformer-VAE with AdamW (lr $= 1\text{e}{-4}$, weight decay $= 1\text{e}{-5}$) and a cosine-to-zero schedule. All checkpoints are from Hugging Face; further details are in the supplement.

4.1. Homogeneous Model Merging

Self-Merging. We first investigate whether exploring a single model's latent distribution can improve its performance by sampling multiple latent codes from its posterior, merging them, and decoding back to parameter space. As shown in Table 1, self-merging yields an $\approx 4\%$ average improvement over the original model and single-sample

	MMLU	MMLU-pro	HellaSwag	Gsm8k
Gemma-3-4b-it	53.10	20.90	47.40	29.90
VAE	54.10 ± 0.36	20.80 ± 0.20	49.03 ± 0.70	31.27 ± 0.55
LS-Merge	**54.20 ± 0.00**	**21.02 ± 0.03**	**50.10 ± 0.00**	**32.20 ± 0.05**
Gemma-3-1b-it	32.20	7.10	28.70	16.90
VAE	32.60 ± 0.26	7.60 ± 0.56	28.57 ± 0.12	16.77 ± 0.12
LS-Merge	**35.13 ± 0.02**	**10.30 ± 0.20**	**31.16 ± 0.14**	**17.50 ± 0.01**

Table 1. Benchmark scores for pretrained model, VAE, and LS-Merge.

	MMLU	MMLU-pro	HellaSwag	Gsm8k	TruthfulQA	NLGraph	K-Crossword	AbstainQA
Best expert	45.7	14.3	46.6	26.1	32.4	51.7	32.7	-10.8
Base model	48.8	18.1	53.3	6.9	30.1	47.5	28.0	-0.9
Data Merge*	44.5	17.6	52.7	14.3	10.7	42.3	**37.0**	-2.5
Uniform Soup	49.7	19.4	54.0	7.9	31.2	47.5	29.6	-0.1
SLERP(t=0.45)	52.5	18.8	50.4	25.5	28.7	49.8	30.0	-0.2
Greedy Soup	50.8	22.1	54.6	23.9	31.9	52.9	28.0	3.3
Dare-Ties	49.1	18.8	53.7	7.3	28.2	52.8	29.0	1.4
LS-Merge(lerp)	54.7	21.6	58.1	**28.1**	**33.0**	53.1	35.6	2.0
LS-Merge(soup)	**56.0**	**22.2**	**60.1**	24.2	32.0	**56.1**	35.2	**4.0**

Table 2. Results on merging expert LoRA weights in the raw weight space and the latent space.

VAE reconstruction, with larger gains on the smaller model, consistent with tighter capacity constraints. Setup details are provided in Appendix D.3.

Expert Merging. We evaluate our primary application: combining specialized LoRA experts from Feng et al. (2024b). Each expert is encoded, latent representations are merged, and the result is decoded into a single fused model. As shown in Table 2, latent-space fusion consistently outperforms all weight-space baselines, including linear and SLERP interpolation under both uniform and greedy soup. By sampling multiple latent codes per expert before merging, LS-Merge explores the learned parameter distribution rather than relying on a single point estimate, yielding more robust combinations than initialization-sensitive methods such as greedy soup.

Comparison to Representation-Merging Methods. We benchmark against Task Arithmetic (Ilharco et al., 2023) and Activation-Informed Merging (AIM) (Nobari et al., 2025), which use task-vector or activation-informed representations beyond direct parameter interpolation. We merge `Llama-2-13B` models fine-tuned on distinct domains using a single VAE trained on the combined weights; for fair comparison, we use *lm-eval* (Gao et al., 2024). As shown in Table 4, our method achieves performance comparable to AIM and substantially outperforms Task Arithmetic, suggesting that latent weight-space merging can match strong methods that use additional representation-level information.

4.2. Cross-Architecture Merging

Our approach supports merging across architectural gaps in two settings: (i) *intra-family* merging across Gemma sizes and (ii) *cross-family* transfer from Llama to Gemma.

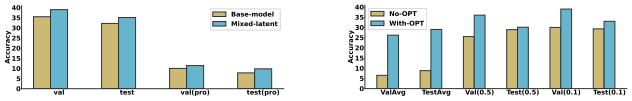
Intra-family (Gemma-3-4B-it → Gemma-3-1B-it). Direct latent interpolation is unstable (Fig. 3b); aligning the larger model's latents to the smaller model's distribution before interpolation yields consistent gains, with small source

Method	MMLU	IFEval	MBPP	HumanEval	GSM8k
base	52.18	25.10	27.80	17.07	4.20
code	52.91	26.25	31.60	17.07	24.10
instruct	53.41	35.67	34.80	26.83	43.40
code + instruct (Task Arithmetic)	52.18	25.10	34.40	26.83	4.20
code + instruct (AIM)	54.18	32.00	36.00	**29.27**	**46.20**
code + instruct (LS-merge)	**55.07**	**36.41**	**36.02**	28.14	44.12

Table 3. Comparison of our latent space merging (LS-merge) with Task Arithmetic and AIM on `Llama-2-13B` fine-tuned models.

Method	MMLU	IFEval	MBPP	HumanEval	GSM8k
base	52.18	25.10	27.80	17.07	4.20
code	52.91	26.25	31.60	17.07	24.10
instruct	53.41	35.67	34.80	26.83	43.40
code + instruct (Task Arithmetic)	52.18	25.10	34.40	26.83	4.20
code + instruct (AIM)	54.18	32.00	36.00	**29.27**	**46.20**
code + instruct (LS-merge)	**55.07**	**36.41**	**36.02**	28.14	44.12

Table 4. Comparison of our latent space merging (LS-merge) with Task Arithmetic and AIM on `Llama-2-13B` fine-tuned models.



(a) Heterogeneous latent space merging.

(b) Varying coefficient on MMLU.

Figure 3. (a) Accuracy gains obtained by latent space merging of `Gemma-3-4B-it` with `Gemma-3-1B-it`. Bars show performance on the validation (*val*) and test splits of the MMLU and MMLU-PRO benchmarks (b) Impact of the merging coefficient t on validation and test accuracy in latent-space merging.

injections ($\lambda \in [0.05, 0.20]$) performing best (Fig. 3a).

Cross-family (Llama-3.2-1B-instruct → Gemma-3-1B-it). Unaligned mixing degrades performance. Table 10 shows that distributionally aligned latent merging recovers and surpasses baselines, with λ=0.1 achieving the best scores. Cross-family results are evaluated with *lm-eval*.

Takeaway. Matching latent dimensionality is insufficient; *aligning latent distributions* is crucial. Once aligned, a single coefficient λ reliably controls capacity transfer from source to target. We report additional results and application studies in Appendix F.

Strategy	WinoGrande	ARC-C	HellaSwag
Base	56.83	42.78	49.07
OT only	51.13	34.25	48.50
OT + interp.	**57.75**	**43.34**	**50.10**

Table 5. Downstream accuracy on WinoGrande, ARC-Challenge, and HellaSwag for different alignment strategies with $\lambda = 0.1$.

Strategy	WinoGrande	ARC-C	MMLU
Base	56.83	42.78	40.76
MLP	56.84	**43.89**	41.02
Attention	56.67	40.23	39.80
Attention + MLP	**57.75**	43.34	**42.10**

Table 6. Ablation: Merging subsets of layers. MLP-only outperforms on ARC-C, combining MLP and Attention achieves the best results.

5. Conclusion

We introduced LS-Merge, a framework that performs model merging in a learned latent space of model parameters. By mapping weights to a continuous manifold and employing optimal transport for principled alignment, our method merges models with heterogeneous architectures, overcoming a fundamental limitation of weight-space techniques. Our experiments demonstrate that LS-Merge excels

at standard expert fusion and enables robust cross-scale and cross-family merging, establishing a scalable, architecture-agnostic paradigm for model composition.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgment

This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)); by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00256259); and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220713, Meta-learning Applicable to Real-world Problems). This research was also supported by a grant from the Korea Machine Learning Ledger Orchestration for Drug Discovery Project (K-MELLODDY), funded by the Ministry of Health & Welfare and the Ministry of Science and ICT, Republic of Korea (Grant No. RS-2024-00460870); and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under the Open RAN Education and Training Program (IITP-2024-RS-2024-00429088), funded by the Korea government (MSIT).

References

Akash, A. K., Li, S., and Trillos, N. G. Wasserstein barycenter-based model fusion and linear mode connectivity of neural networks, 2022. URL https://arxiv.org/abs/2210.06671.

Akiba, T., Shing, M., Tang, Y., Sun, Q., and Ha, D. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, 7(2):195–204, January 2025. ISSN 2522-5839. doi: 10.1038/s42256-024-00975-8. URL http://dx.doi.org/10.1038/s42256-024-00975-8.

Bengio, Y., Courville, A. C., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35: 1798–1828, 2012.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ding, W., Feng, S., Liu, Y., Tan, Z., Balachandran, V., He, T., and Tsvetkov, Y. Knowledge crosswords: Geometric knowledge reasoning with large language models, 2024. URL https://arxiv.org/abs/2310.01290.

Eckart, C. and Young, G. M. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

Feng, S., Shi, W., Wang, Y., Ding, W., Balachandran, V., and Tsvetkov, Y. Don't hallucinate, abstain: Identifying LLM knowledge gaps via multi-LLM collaboration. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, August 2024a.

Feng, S., Wang, Z., Wang, Y., Ebrahimi, S., Palangi, H., Miculicich, L., Kulshrestha, A., Rauschmayr, N., Choi, Y., Tsvetkov, Y., Lee, C.-Y., and Pfister, T. Model swarms: Collaborative search to adapt llm experts via swarm intelligence, 2024b. URL https://arxiv.org/abs/2410.11163.

Flamary, R., Courty, N., Gramfort, A., Alaya, M. Z., Boisbunon, A., Chambon, S., Chapel, L., Corenflos, A., Fatras, K., Fournier, N., Gautheron, L., Gayraud, N. T., Janati, H., Rakotomamonjy, A., Redko, I., Rolet, A., Schutz, A., Seguy, V., Sutherland, D. J., Tavenard, R., Tong, A., and Vayer, T. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

Flamary, R., Vincent-Cuaz, C., Courty, N., Gramfort, A., Kachaiev, O., Quang Tran, H., David, L., Bonet, C., Cassereau, N., Gnassounou, T., Tanguy, E., Delon, J., Collas, A., Mazelet, S., Chapel, L., Kerdoncuff, T., Yu, X., Feickert, M., Krzakala, P., Liu, T., and Fernandes Montesuma, E. Pot python optimal transport (version 0.9.5), 2024. URL https://github.com/PythonOT/POT.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Gururangan, S., Li, M., Lewis, M., Shi, W., Althoff, T., Smith, N. A., and Zettlemoyer, L. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. Lorahub: Efficient cross-task generalization via dynamic lora composition, 2023.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Jang, D.-H., Yun, S., and Han, D. Model stock: All we need is just a few fine-tuned models. In *Proceedings of the European Conference on Computer Vision*, 2024.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL https://api.semanticscholar.org/CorpusID:216078090.

Knyazev, B., Moudgil, A., Lajoie, G., Belilovsky, E., and Lacoste-Julien, S. Accelerating training with neuron interaction and nowcasting networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cUFIil6hEG.

Lahiri, S., Gao, P., and Ganguli, S. Random projections of random manifolds, 2016. URL https://arxiv.org/abs/1607.04331.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

llam 3 team. The llama 3 herd of models, 2024.

Mavromatis, C., Karypis, P., and Karypis, G. Pack of llms: Model fusion at test-time via perplexity optimization. *ArXiv*, abs/2404.11531, 2024. URL https://api.semanticscholar.org/CorpusID:269188153.

Navon, A., Shamsian, A., Achituve, I., Fetaya, E., Chechik, G., and Maron, H. Equivariant architectures for learning in deep weight spaces. *arXiv preprint arXiv:2301.12780*, 2023.

Nobari, A. H., Alimohammadi, K., ArjomandBigdeli, A., Srivastava, A., Ahmed, F., and Azizan, N. Activation-informed merging of large language models, 2025. URL https://arxiv.org/abs/2502.02421.

Peebles, W., Radosavovic, I., Brooks, T., Efros, A. A., and Malik, J. Learning to learn with generative models of neural network checkpoints, 2022. URL https://arxiv.org/abs/2209.12892.

Pennec, X. Barycentric subspace analysis on manifolds, 2017. URL https://arxiv.org/abs/1607.02833.

Santambrogio, F. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*, volume 87 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser, 2015.

Sch"urholt, K., Mahoney, M. W., and Borth, D. Towards scalable and versatile weight space learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Si, C., Jiang, J., and Shen, W. Unveiling the mystery of weight in large foundation models: Gaussian distribution never fades, 2025.

Team, G. et al. Gemma 3 technical report, 2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Villani, C. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language?, 2024a. URL https://arxiv.org/abs/2305.10037.

Wang, K., Tang, D., Zhao, W., Schürholt, K., Wang, Z., and You, Y. Recurrent diffusion for large-scale parameter generation, 2025.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024b.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. 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*, pp. 23965–23998, 2022.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

Yang, E., Shen, L., Guo, G., Wang, X., Cao, X., Zhang, J., and Tao, D. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities, 2024.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *International Conference on Machine Learning*, 2024b.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zhou, A., Yang, K., Burns, K., Cardace, A., Jiang, Y., Sokota, S., Kolter, J. Z., and Finn, C. Permutation equivariant neural functionals. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=fmYmXNPmhv.

Impact Statement

We introduce a VAE-based latent-space merging technique for pretrained LLMs that encodes weights into compact codes, blends them, and decodes new high-performance parameters. This enables fast, compute-efficient model customization and smooth interpolation across architectures. However, blending latent codes can unintentionally merge biases or toxic behaviors, obscure the origin of capabilities, and be misused to graft malicious functionality. We advocate for rigorous bias/toxicity audits, transparent provenance tracking, and clear reporting guidelines to ensure responsible deployment.

Limitations. Despite its strong performance, training the weight-encoding VAE at high compression ratios can be challenging due to mode collapse, largely caused by the heavy-tailed nature of LLM weight distributions. Nonetheless, our merging paradigm does not strictly require a tight bottleneck; it remains highly effective with an overcomplete latent space, which eases optimization while successfully unfolding the weight manifold.

This work opens several avenues for future research, including developing more efficient, property-aware weight encoders (Navon et al., 2023; Zhou et al., 2023) to improve scalability, and extending our framework beyond simple merging via iterative latent barycentric interpolation (Akash et al., 2022; Pennec, 2017) to unlock complex uni-model compositions.

Reproducibility Statement

To ensure reliable and reproducible results, we provide detailed experimental settings in the Appendix. We plan to open-source our implementation.

A. Weight Distribution Analysis

To motivate our encoder design, we analyze the distributional properties of transformer submodules in Gemma-3 (Team et al., 2025) and LLaMA-3.2 (llam 3 team, 2024). We compute the first four moments (mean, variance, skewness, and kurtosis) for self-attention layers (q_proj, k_proj, o_proj) and MLP layers (up_proj, down_proj, gate_proj). Table 7 reports per-moment summaries; full per-layer statistics are in Table 8.

Weights exhibit near-zero means, low variances, and small positive skewness, but *markedly high kurtosis* in earlier layers, especially in self-attention projections. High kurtosis (leptokurtic distributions) indicates sharp peaks with heavy tails: rare, large-magnitude parameters that are likely functionally important. This contradicts Gaussian assumptions used in prior work (Si et al., 2025) and suggests that encoders must preserve tail events rather than over-regularize toward a narrow Gaussian. We observe this pattern consistently across sizes within both families.

layers	llama3-2-3b-it				gemma-3-1b-it				gemma-3-4b-it			
	mean	var.	skew	kurt.	mean	var.	skew	kurt.	mean	var.	skew	kurt.
self_attn	0	0.0017	0.0192	8.4032	0.0002	0.0031	0.0500	15.0505	0	0.0015	0.0501	15.2010
	0	0.0013	0.0191	7.3438	0.0002	0.0030	0.0450	9.8347	0	0.0015	0.0267	7.3832
	0	0.0012	0.0178	6.2172	0.0001	0.0030	0.0389	9.0363	0	0.0014	0.0255	7.2277
	0	0.0011	0.0104	5.4477	0.0001	0.0030	0.0288	8.6496	0	0.0014	0.0172	6.0731
avg (self_attn)	0	0.0005	-0.0002	1.4342	0	0.0012	-0.0009	3.2858	0	0.0005	0.0009	2.6900
mlp	0	0.0005	0.0093	5.4740	0.0002	0.0010	0.0167	8.7665	0	0.0004	0.0266	6.3297
	0	0.0005	0.0080	4.1465	0.0002	0.0010	0.0090	3.1514	0	0.0004	0.0155	5.9670
	0	0.0004	0.0075	2.6694	0.0002	0.0010	0.0077	3.0577	0	0.0004	0.0130	2.7454
	0	0.0004	0.0074	2.4364	0.0002	0.0010	0.0071	3.0113	0	0.0004	0.0094	2.4371
avg (mlp)	0	0.0003	0.0003	0.8435	0	0.0006	0	1.1739	0	0.0003	0.0006	1.0807

Table 7. Layer-wise distribution statistics of model parameters for three instruction-tuned LLMs: **llama3-2-3b-it**, **gemma-3-1b-it**, and **gemma-3-4b-it**. For each model and block type (`self_attn` and `mlp`), we report the first four moments of the flattened weight tensors: *mean*, *variance*, *skewness*, and *excess kurtosis*. Per-row entries list representative layers in each block; the `avg` row aggregates across all layers for that block and model. Means are near zero and variances are small, while skewness and especially kurtosis (often > 5 and up to ~ 15) indicate pronounced heavy tails and asymmetry. These non-Gaussian, heavy-tailed statistics motivate encoders that preserve rare but high-magnitude outliers.

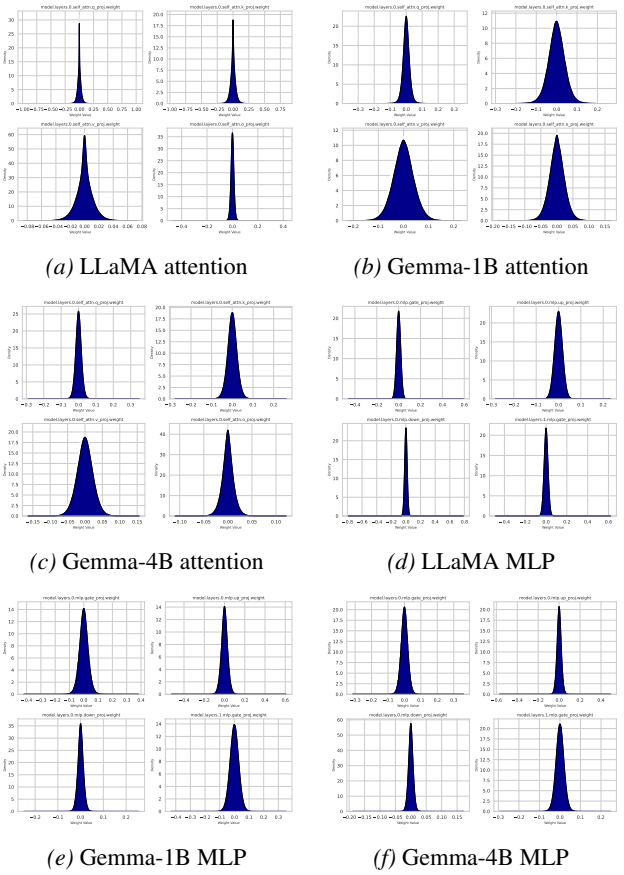
Per-Layer Analysis. We provide complete per-layer distribution plots for all MLP layers of GEMMA-3-1B-IT, GEMMA-3-4B-IT, and LLAMA-3.2-3B-INSTRUCT, along with cumulative variance-explained curves from PCA. For each transformer block, we analyze self-attention (Q/K/V/O projections) and MLP (up/gate/down) weight matrices independently. Statistics are computed on raw matrix entries; for PCA, matrices are mean-centered and the spectrum is computed over flattened rows. We use excess kurtosis (Fisher convention) so that a Gaussian has κ_{ex}=0:

$$\kappa_{\text{ex}} = \frac{\mathbb{E}[(W-\mu)^4]}{\sigma^4} - 3.$$

Family summaries first average across layers within a model, then across models in the same family.

Self-attention projections in GEMMA show pronounced positive excess kurtosis with depth-localized peaks (Fig. 6a, Table 8), indicating heavy tails and extreme outliers. LLAMA models track closely across scales, suggesting family-level statistical stability. MLP projections are near-Gaussian with excess kurtosis typically in $[0, 2]$ and fewer depth-dependent spikes (Fig. 6b). Their PCA spectra (Fig. 5) decay faster than attention, indicating lower intrinsic dimensionality.

Design Implications. Heavy tails in attention (notably in Gemma) warrant encoders with higher capacity or robust priors, as Gaussian assumptions under-represent extremes. The more stable, near-Gaussian MLP statistics admit accurate compression with standard VAE settings. The consistency of LLAMA statistics across scales simplifies cross-scale latent calibration and reduces merging friction.

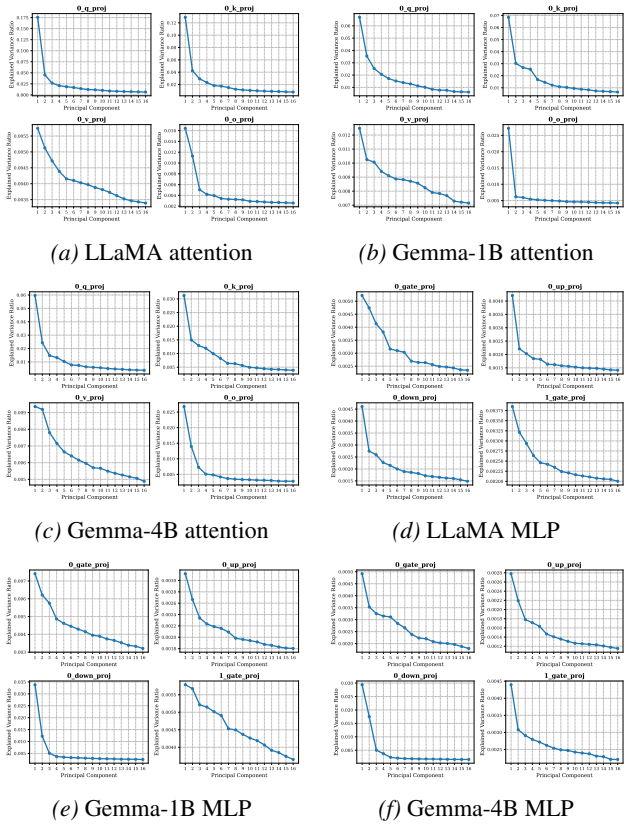


(a) LLaMA attention *(b)* Gemma-1B attention

(c) Gemma-4B attention *(d)* LLaMA MLP

(e) Gemma-1B MLP *(f)* Gemma-4B MLP

Figure 4. Weight distributions of self-attention and MLP modules for LLaMA-3.2-3B-IT, Gemma-3-1B-IT, and Gemma-3-4B-IT at layer 0.



(a) LLaMA attention *(b)* Gemma-1B attention

(c) Gemma-4B attention *(d)* LLaMA MLP

(e) Gemma-1B MLP *(f)* Gemma-4B MLP

Figure 5. PCA explained-variance ratios for individual weight matrices in the first self-attention and MLP blocks of LLaMA-3.2-3B-IT, Gemma-3-1B-IT, and Gemma-3-4B-IT. The sharp decay after the leading principal components indicates a pronounced low-rank structure across architectures and model sizes.

A.1. Theoretical Compressibility of LLM Weights

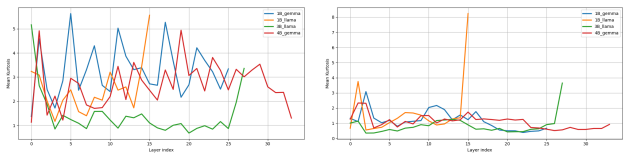
Given the heavy tails and low intrinsic variance observed above, we investigate whether a VAE can compress LLM weights effectively.

Let $W \in \mathbb{R}^{n\times m}$ be a layer matrix and $w = \mathrm{vec}_{\mathrm{row}}(W) \in \mathbb{R}^D$ its row-wise flattening with $D = nm$. Empirically (Fig. 5), the top $r \ll \min(n, m)$ principal components capture nearly all variance: $\sum_{i=1}^{r} \lambda_i / \sum_{i=1}^{D} \lambda_i \approx 1$, where $\{\lambda_i\}$ are the eigenvalues of the empirical covariance of w. By the Eckart–Young theorem (Eckart & Young, 1936), the best rank-r approximation W_r satisfies $\|W - W_r\|_F^2 = \sum_{i>r} \lambda_i$ and requires $O(r(n + m))$ parameters. Thus, weights concentrate near a low-dimensional set.

Assuming the collection of flattened weights $\mathcal{W} \subset \mathbb{R}^D$ concentrates near a smooth d-dimensional manifold $\mathcal{M}$ with $d \ll D$, manifold embedding results (Bengio et al., 2012; Lahiri et al., 2016) imply the existence of $\Phi : \mathbb{R}^D \to \mathbb{R}^k$ with $k = O\big(\frac{d}{\varepsilon^2} \log \frac{V}{\tau}\big) \ll D$ that preserves pairwise distances on $\mathcal{M}$ within $(1 \pm \varepsilon)$, where V bounds manifold volume and τ its reach. A VAE encoder $f_\theta : \mathbb{R}^D \to \mathbb{R}^k$ with decoder $g_\phi : \mathbb{R}^k \to \mathbb{R}^D$ can approximate such compressive embeddings while modeling uncertainty, justifying our latent compression approach.

B. Optimal Transport Derivation

We formalize heterogeneous latent alignment as the Monge problem under the squared Euclidean transport cost. Given



(a) Mean kurtosis per layer (Self-attention layers). *(b)* Mean kurtosis per layer (MLP layers).

Figure 6. Comparison of mean kurtosis across layers in self-attention and MLP components for Gemma and LLaMA models.

layers	llama3-2-3b-it			gemma-3-1b-it			gemma-3-4b-it		
	var.	skew	kurt.	var.	skew	kurt.	var.	skew	kurt.
self_attn	0.0017	0.0192	8.4032	0.0031	0.0500	15.0505	0.0015	0.0501	15.2010
	0.0013	0.0191	7.3438	0.0030	0.0450	9.8347	0.0015	0.0267	7.3832
	0.0012	0.0178	6.2172	0.0030	0.0389	9.0363	0.0014	0.0255	7.2277
	0.0011	0.0104	5.4477	0.0030	0.0288	8.6496	0.0014	0.0172	6.0731
avg (self_attn)	0.0005	-0.0002	1.4342	0.0012	-0.0009	3.2858	0.0005	0.0009	2.6900
min (self_attn)	0.0001	-0.0131	0.3027	0.0002	-0.0418	0.1589	0.0001	-0.0305	0.2412
mlp	0.0005	0.0093	5.4740	0.0010	0.0167	8.7665	0.0004	0.0266	6.3297
	0.0005	0.0080	4.1465	0.0010	0.0090	3.1514	0.0004	0.0155	5.9670
	0.0004	0.0075	2.6694	0.0010	0.0077	3.0577	0.0004	0.0130	2.7454
	0.0004	0.0074	2.4364	0.0010	0.0071	3.0113	0.0004	0.0094	2.4371
avg (mlp)	0.0003	0.0003	0.8435	0.0006	0	1.1739	0.0003	0.0006	1.0807
min (mlp)	0.0003	-0.0139	0.0892	-0.0001	-0.0184	0.1552	0	-0.0076	0.1559

Table 8. Statistical moments of the self-attention and MLP layers across three models.

source and target latent distributions μ_{src} and μ_{tgt}, we seek

$$T^* = \arg\min_T \int \|z - T(z)\|_2^2 \, d\mu_{\text{src}}(z) \quad \text{s.t.} \quad T_{\#}\mu_{\text{src}} = \mu_{\text{tgt}}, \tag{8}$$

where $T_{\#}\mu_{\text{src}}$ denotes the pushforward measure induced by T.

The general Monge problem is computationally intensive. We approximate each layer's latent distribution as a Gaussian with empirical mean and covariance. Let the source and target statistics be (μ_s, Σ_s) and (μ_t, Σ_t), respectively. Under this approximation, the optimal map admits the affine form

$$\tilde{z}_{\text{src}} = T^*(z_{\text{src}}) = \mu_t + A(z_{\text{src}} - \mu_s), \tag{9}$$

where

$$A = \Sigma_s^{-1/2} \left(\Sigma_s^{1/2} \, \Sigma_t \, \Sigma_s^{1/2} \right)^{1/2} \Sigma_s^{-1/2}. \tag{10}$$

This transformation aligns the source mean and covariance with the target latent distribution, producing transported source latents compatible with the target decoder.

C. Experimental Setup Details

C.1. Datasets and Tasks

We use the evaluation suite of Feng et al. (2024b): language understanding (MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024b)), commonsense reasoning (HellaSwag (Zellers et al., 2019)), math (GSM8k (Cobbe et al., 2021)), and knowledge-intensive tasks (Knowledge Crosswords (Ding et al., 2024), NLGraph (Wang et al., 2024a), TruthfulQA (Lin et al., 2022), AbstainQA (Feng et al., 2024a)). Training data consist of pretrained weight snapshots for `Gemma-3-1B-it` and `Gemma-3-4B-it`, plus LoRA experts from Feng et al. (2024b).

C.2. Baselines

We compare against reference-free weight-space methods: spherical linear interpolation (SLERP), Uniform Soup (Wortsman et al., 2022), Greedy Soup (Wortsman et al., 2022), data-level merging, raw pretrained checkpoints, and the best single expert per task. Methods requiring an unmodified base reference are excluded.

C.3. Computational Cost

All experiments were conducted on a single NVIDIA RTX 6000 Ada GPU. Training a $\sim$200M parameter VAE at compression ratio r=1.6 requires approximately 1–2 hours for 1B-scale models and 3–4 hours for 7B-scale models ($\sim$500K chunks). Higher compression ratios increase training time proportionally. LoRA weights converge notably faster due to their lower-rank structure. At inference, the full encode-decode pipeline completes in 5–10 seconds, enabling rapid iteration during merging experiments.

D. VAE Architecture, Training, and Convergence

Table 9 lists the configuration of our Transformer-VAE. The encoder applies a twofold down-projection (compression ratio r=2), the smallest bottleneck that preserves reconstruction while generalizing to unseen checkpoints. The decoder mirrors the encoder to restore the original dimension. For heterogeneous merges, we set the latent size d_z by the source-to-target layer mapping ratio and keep all other hyperparameters fixed, ensuring comparable latent scales across architectures.

Objective. We optimize the ELBO (reconstruction + KL to $\mathcal{N}(0, I)$) with a constant learning rate 1×10^{-5} for 10,000 epochs.

Training Setup. Single NVIDIA A6000, bfloat16 precision for forward and backward passes. Remaining settings (optimizer, batch size, weight decay, gradient clipping) are in Table 9. Seeds are fixed for data order and initialization.

Field	Value	Notes
`length`	10,240	Sequence length processed
`n_layers`	6	Transformer depth
`chunk_size`	640	Per-token chunk width
`embed_dim`	768	Matches global `embed_dim`
`latent_dim`	640	Size of latent vector per chunk
`n_heads`	4	Attention heads
`rope_base`	20,000	RoPE base frequency
`conv`	`false`	No convolutional patching
`flatten`	`true`	Flatten to original shape.

Table 9. Transformer block settings shared by the encoder and decoder.

D.1. Overcomplete Latent Spaces

For heterogeneous models, expansion (r<1) can offer an alternative trade-off. Lifting weights into an overcomplete

latent space unfolds curved manifolds, allowing linear interpolation to better approximate geodesics on the original weight manifold. Expansion also forces the encoder to learn a non-trivial transformation, avoiding identity collapse. Practically, the relaxed bottleneck accelerates training and improves gradient flow at the cost of increased memory.

Two-Stage Curriculum. Since LLM weight distributions have low variance and heavy tails (Section A; Fig. 4), training VAEs directly on these weights can lead to early collapse. We stabilize training with (i) transformer blocks (Vaswani et al., 2017) in E_θ, D_ϕ for long-range coupling across chunks, and (ii) a two-stage curriculum: first train a deterministic autoencoder (KL off) to convergence, then enable the KL term and fine-tune to structure the latent space without sacrificing fidelity. This prevents posterior collapse and yields stable convergence even at high compression ratios, without increasing total training time. VAE performance is measured by the downstream accuracy of the reconstructed weights when used to initialize the corresponding architectures.

Ablation Summary. Bottlenecks weaker than $r{=}2$ overfit and degrade out-of-distribution reconstruction; stronger bottlenecks raise reconstruction error and harm downstream merging. Setting d_z via the mapping ratio stabilizes cross-architecture alignment without retuning.

D.2. Convergence Analysis

We study how the compression ratio r interacts with the variability in tail behavior across layers (cf. Section A).

Convergence vs. Compression. Figure 7 shows a clear monotone effect: optimization slows as the bottleneck tightens. At $r{=}2$, storage is halved and reconstruction loss increases by only $\approx 6\%$; at $r{=}4$, the penalty rises to $\approx 10\%$. These curves set a practical operating point: $r{\approx}2$ balances footprint and fidelity with minimal training overhead.

Sensitivity to Distribution Shape. Layers with heavier tails (high excess kurtosis) are more sensitive to increasing r, exhibiting larger loss gaps at fixed training budget; near-Gaussian layers degrade more gently. Attention projections in GEMMA fall into the former regime, while MLP projections across families align with the latter (Section A).

D.3. Self-Merging Setup

We use a single Transformer-VAE with six encoder and decoder blocks, trained jointly on weights from `Gemma-3-1B-it` and `Gemma-3-4B-it` with compression ratio 2.

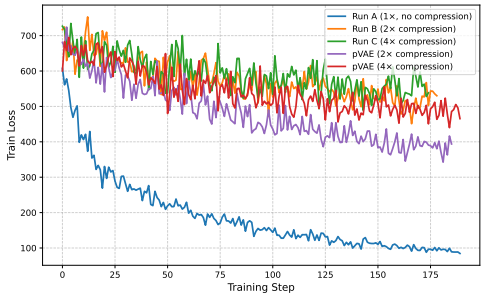


Figure 7. Mean reconstruction error curves (smoothed with a 5-step moving average) for pVAE (VAE initialized from an existing pretrained VAE with reduction factor 1) and VAE (random initialized VAE). r denotes the compression ratio.

E. Heterogeneous Model Merging in Latent Space

Merging models with matched layer shapes is straightforward in weight space. Heterogeneous pairs are harder: to avoid truncating the higher-capacity model, we train separate VAEs per architecture and merge in a shared, aligned latent space. Concretely, we calibrate latents per layer, fit an alignment map between the source and target latent spaces, and interpolate on the target side before decoding. This preserves each model's capacity while enabling stable cross-architecture merges without ad hoc dimensionality cuts. The full procedure is given in Algorithm 1.

E.1. Intuition Behind Latent Space Alignment

The rationale for restricting explicit alignment to heterogeneous merging relies on the geometric properties of the loss landscape and the topological structure of parameter manifolds.

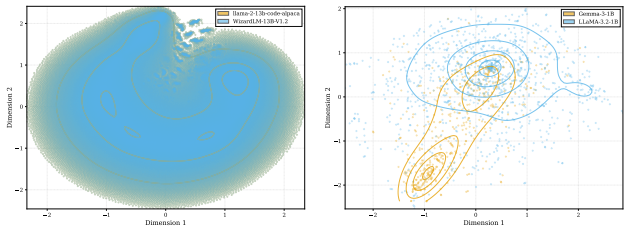
Homogeneous Merging and Linear Mode Connectivity. For homogeneous models finetuned from a shared initialization (e.g., distinct fine-tunes of `LLaMA-2-13B`), the parameters remain within a shared basin of attraction. Their weight vectors θ_1, θ_2 reside on a connected region of the parameter manifold $\mathcal{M}$. When encoded into the latent space $\mathcal{Z}$, the resulting distributions $p(z|\theta_1)$ and $p(z|\theta_2)$ naturally share common support and overlapping density. Interpolation in $\mathcal{Z}$ therefore corresponds to traversing a flat, low-loss path on the underlying manifold, rendering explicit distributional alignment redundant.

The Heterogeneous Disconnect. Heterogeneous models (e.g., Gemma-3-1B vs. Llama-3.2-1B) possess fundamentally distinct architectures. Even when projected into a latent space of identical dimensionality d, their representations occupy disjoint manifolds $\mathcal{M}_{\text{src}}$ and $\mathcal{M}_{\text{tgt}}$ with divergent geometric structures and density profiles (Fig-

Algorithm 1 Latent-Space Model Merging

Require: W_{src}, W_{tgt}; encoders E_s, E_t; decoder D_t; types $\mathcal{T}$; per-chunk dim d; λ
Ensure: W_{merged}
for each layer type $t \in \mathcal{T}$ **do**
  $u_s^t \leftarrow \text{concat}\big(E_s\big(\text{chunk}(w_{\text{src}}^t)\big)_j\big)$ ▷ $\in \mathbb{R}^{L_s^t}$
  $u_t^t \leftarrow \text{concat}\big(E_t\big(\text{chunk}(w_{\text{tgt}}^t)\big)_j\big)$ ▷ $\in \mathbb{R}^{L_t^t}$
  Dimension alignment $r \leftarrow \frac{n_t^t\, N_{\text{tgt}}}{n_s^t\, M_{\text{src}}}$ **so** $K_s^t = K_t^t = K$
  $Z_s^t \leftarrow \text{split}(u_s^t,\, d) \in \mathbb{R}^{K \times d}$
  $Z_t^t \leftarrow \text{split}(u_t^t,\, d) \in \mathbb{R}^{K \times d}$
  $(\mu_s^t,\, \Sigma_s^t),\ (\mu_t^t,\, \Sigma_t^t) \leftarrow$ stats of rows
  $A^t \leftarrow \Sigma_s^{-1/2}\big(\Sigma_s^{1/2}\, \Sigma_t\, \Sigma_s^{1/2}\big)^{1/2}\Sigma_s^{-1/2}$ ▷ OT push
  $\tilde{Z}_s^t \leftarrow \mu_t^t + (Z_s^t - \mu_s^t)(A^t)^\top$
  $Z_\lambda^t \leftarrow (1-\lambda)\, Z_t^t + \lambda\, \tilde{Z}_s^t$
  $Z_\lambda^{t,\text{dec}} \leftarrow \text{reshape}(Z_\lambda^t,\, \text{shape}_{D_t})$
  $\hat{W}^t \leftarrow D_t(Z_\lambda^{t,\text{dec}})$
  Flatten decoder output & unchunk $\hat{W}^t$ into target layers of type t
end for
return W_{merged}

ure 8). This disjoint nature necessitates a mapping to bridge $\mathcal{M}_{\text{src}}$ and $\mathcal{M}_{\text{tgt}}$. We employ Optimal Transport (specifically the Monge map) to push μ_{src} onto μ_{tgt} by minimizing the Wasserstein-2 distance. This registers the two manifolds, aligning their statistical moments and geometric structure, ensuring that interpolated latent codes z_α remain within the valid density region of the target decoder.



(a) Same-Architecture Fine-tuning Variants: LLaMA-2-13B Code-Alpaca vs WizardLM-13B-V1.2.

(b) Cross-Architecture Comparison: Gemma-3-1B vs LLaMA-3.2-1B.

Figure 8. **VAE latent space reveals architectural signatures in weight distributions.** t-SNE projections of encoded k_proj weights show (**a**) fine-tuned variants of the same base model (LLaMA-2-13B) remain indistinguishable while (**b**) distinct clusters for different architectures (Gemma vs LLaMA) , suggesting the learned latent space captures intrinsic architectural properties.

F. Ablation Studies

We conduct targeted ablation studies to understand the properties of our method. Unless stated otherwise, we use *lm-eval* for evaluation.

F.1. Component Contributions in Latent Merging

We analyze how different submodules contribute by merging MLP layers only, self-attention layers only, or both jointly. The results in Table 11 show that merging MLP layers alone provides modest gains, while merging attention layers alone degrades performance. Optimal results require merging both, indicating that **MLP and self-attention parameters encode complementary functional knowledge**, and that altering one without the other disrupts learned co-adaptations.

F.2. VAE Generalization and the Compression Trade-off

We assess zero-shot generalization by training the VAE on `Gemma-3-4B-it` and evaluating on two unseen models: in-family `Gemma-3-1B-it` and out-of-family `Llama-3.2-1B-it`. Table 12 reveals a clear trade-off: at low compression (r=1.6) the VAE maintains strong performance on both unseen models, but performance degrades substantially at higher ratios (r=2, 4). This suggests that while the VAE learns transferable weight representations, aggressive compression induces posterior collapse as most weight values cluster near zero.

Strategy	WinoGrande	ARC-C	HellaSwag
Base	56.83	42.78	49.07
OT only	51.13	34.25	48.50
OT + interp.	**57.75**	**43.34**	**50.10**

Table 10. Downstream accuracy on WinoGrande, ARC-Challenge, and HellaSwag for different alignment strategies with $\lambda = 0.1$.

Strategy	WinoGrande	ARC-C	MMLU
Base	56.83	42.78	40.76
MLP	56.84	**43.89**	41.02
Attention	56.67	40.23	39.80
Attention + MLP	**57.75**	43.34	**42.10**

Table 11. Ablation: Merging subsets of layers. MLP-only outperforms on ARC-C, combining MLP and Attention achieves the best results.

F.3. Linear Subspace vs. Non-Linear Manifold Learning

Although Section A showed that LLM weight matrices exhibit low-rank structure, this does not imply that functional parameters form a linear subspace. We compare our Transformer-VAE against incremental PCA on `Gemma-3-1B-it` across compression ratios $r \in \{1.6, 2.0, 4.0\}$ (Table 13).

PCA Collapses Functional Performance. Across all ratios, PCA-reconstructed models regress to near-random accuracy on MMLU ($\approx$25.5% at r=1.6) with global drops across benchmarks (e.g., ARC-C: 42.41% $\rightarrow$ 27.65%). Performance is equally poor at r=1.6 and r=4.0, indicating a structural mismatch rather than insufficient capacity: the set of valid pretrained weights does not reside in a linear subspace.

Model	Winogrande (5-shot)	ARC-Challenge (25-shot)	HellaSwag (10-shot)	MMLU (5-shot)
Gemma-3-1B-it (base)	56.83	42.78	49.07	40.76
LLaMA-3.2-1B-it (base)	61.56	41.11	61.62	46.55
Gemma-3-1B-it (VAE, $r = 1.6$)	56.67	42.83	47.31	39.98
LLaMA-3.2-1B-it (VAE, $r = 1.6$)	61.25	41.47	60.80	46.06
Gemma-3-1B-it (VAE, $r = 2$)	56.67	38.23	38.88	32.22
LLaMA-3.2-1B-it (VAE, $r = 2$)	59.43	37.12	55.99	39.73
Gemma-3-1B-it (VAE, $r = 4$)	46.49	28.24	25.66	25.02
LLaMA-3.2-1B-it (VAE, $r = 4$)	49.20	26.28	25.70	26.76

Table 12. Accuracy of VAE models (trained on `Gemma-3-4B-it`) when evaluated on `Gemma-3-1B-it` and `LLaMA-3.2-1B-it`. Performance remains stable for $r = 1.6$, but deteriorates as the compression factor increases.

Table 13. Functional Reconstruction Fidelity vs. Compression Ratio. Zero-shot accuracy comparison on standard benchmarks (MMLU, HellaSwag, Winogrande, ARC-C) for Gemma-3-1B-it. We compare our non-linear Transformer-VAE (LS-Merge) against a linear PCA baseline at compression ratios $r \in \{1.6, 2, 4\}$. While the VAE maintains strong performance at $r = 1.6$ and $r = 2$, linear compression suffers significantly as the bottleneck tightens, validating the need for non-linear manifold learning.

Method	Ratio (r)	MMLU	HellaSwag	Winogrande	ARC-C
Gemma-3–1b-it)	1.0×	41.44 ± 0.00	49.05 ± 0.01	55.41 ± 0.03	42.41 ± 0.02
PCA (Linear)	1.6×	25.50 ± 0.37	25.56 ± 0.04	50.12 ± 0.01	27.65 ± 0.01
LS-Merge VAE	1.6×	**39.89** ± 0.01	**48.57** ± 0.25	**56.64** ± 0.15	**41.64** ± 0.01
PCA (Linear)	2.0×	24.12 ± 0.00	25.27 ± 0.12	46.27 ± 0.01	26.24 ± 0.32
LS-Merge VAE	2.0×	**39.80** ± 0.00	**49.29** ± 0.10	**54.14** ± 1.02	**42.32** ± 0.21
PCA (Linear)	4.0×	24.13 ± 0.15	24.79 ± 0.23	49.57 ± 0.01	25.89 ± 0.23
LS-Merge VAE	4.0×	**39.83** ± 0.00	**49.30** ± 0.21	**56.06** ± 0.15	**42.75** ± 0.20

VAE Preserves the Functional Manifold. The LS-Merge VAE retains near-original accuracy at all compression levels. At r=1.6, it recovers 96% of the base model's MMLU performance (39.89% vs. 41.44%) and even improves Winogrande (56.64% vs. 55.41%). VAE reconstructions remain stable at r=4.0, where PCA has already collapsed at r=1.6. This confirms that pretrained weights lie on a non-linear manifold requiring expressive encoders and decoders to model its curvature. Linear projections cannot preserve the structure of the weight manifold and produce functionally invalid models even under mild compression. The VAE's non-linear latent geometry is therefore a geometric necessity for faithful reconstruction and downstream merging.