Debugging ReBasin: What Limits Symmetry-Based Model Merging?

Elliot Stein [1] **Christine Evers** [1] **Jonathon Hare** [1]

Abstract

Poor weight-space model-merging accuracy is often taken as evidence that models have not been well aligned, motivating further development of alignment methods. However, weights-merge accuracy conflates alignment quality with architectural constraints, folding error, and normalisation. We introduce a representation-merge diagnostic that applies the same alignment maps directly at runtime, then progressively constrains this idealised merge to isolate contributors to the final weights-merge result. On CIFAR-100 with VGG-16-BN and ResNet-20, this decomposition reveals a structural cost in narrow-ResNet merging that is irreducible to alignment quality: one residual-stream alignment must be reused at multiple positions within a stage, a constraint of architecture rather than of any particular merge method. Trained alignment maps outperform closed-form alignments at narrow width, but this gap closes as width grows. Together, these results show that symmetry-based merging is limited both by the ability to find good alignments and by structural constraints on how those alignments are realised within architectures. Understanding each is important for progress in model merging.

1. Introduction

Independently trained networks that achieve similar test accuracy can occupy distant points in weight space because of symmetries in their parameterisation. The linear mode connectivity programme (Entezari et al., 2022) and the rebasin literature it has motivated (Ainsworth et al., 2023; Singh & Jaggi, 2020; Peña et al., 2023; Akash et al., 2022) have shown that, in sufficiently overparameterised settings, models from different initialisations can often be aligned by permutations of their hidden units and connected by low-loss paths or merged with low loss. For narrower architectures, especially residual ones, permutation-based merging degrades sharply (Ainsworth et al., 2023; Jordan et al., 2023), motivating richer alignment families: canonical-correlation-analysis (CCA)-style projections (Horoi et al., 2024), greedy many-to-many assignment (Stoica et al., 2024; Singh & Jaggi, 2020), permutation matching with layer-wise least-squares optimisation (Nasery et al., 2025), autoencoder-based parameter canonicalisation (Boufalis et al., 2025), and gradient-based methods (Kinderman et al., 2025). Recent analyses also study regimes in which permutation alignment becomes sufficient as width grows (Ito et al., 2025b;a); our experiments operate at finite practical widths where this asymptotic picture need not yet hold.

Because these methods are typically compared by weights-merge accuracy, poor merge performance is naturally interpreted as poor alignment. But this metric mixes several distinct costs: the quality of the alignment itself, the structural cost of reusing the alignment across architectural positions, the cost of folding it algebraically into parameters, and post-merge normalisation. Recent work has begun to separate alignment from post-hoc statistics correction (Sharma et al., 2024; Jordan et al., 2023); we extend this decomposition by additionally isolating structural and folding costs. Without separating these, the central question motivating richer alignment methods is not directly testable: are failures caused by inadequate alignments, or by downstream costs imposed when those alignments are realised inside a particular architecture?

We disentangle these costs by comparing weight-space merging against the same alignment maps applied directly at runtime as representation-space averaging. Linear encoder–decoder pairs are inserted at alignment boundaries, mapping each model's private representations to a shared latent space; the maps can be obtained by any closed-form or trained alignment method. Representation-merge averages activations in the latent space, while weights-merge folds the encoders and decoders into surrounding parameters before averaging. For sequential architectures, the comparison is direct. For residual architectures, where a single residual-stream alignment must be folded into multiple parameter locations, we introduce a staircase of representation-merge variants that progressively constrain the idealised merge to mirror the structure weights-merge imposes.

[1]University of Southampton. Correspondence to: Elliot Stein <e.stein@soton.ac.uk>.

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).

Two findings emerge. First, narrow-ResNet merging incurs a structural cost that is irreducible to alignment quality or algebraic folding: the requirement that one residual-stream alignment serve multiple positions within a residual stage, a constraint imposed by the architecture. Second, closed-form alignment methods, including permutation matching and CCA-style maps, saturate below trained encoder–decoder maps at narrow width, with the gap closing as width grows. This width-dependent gap is consistent with narrow models using more polysemantic, entangled representations (Elhage et al., 2022; Bricken et al., 2023), making alignment less reducible to permutation matching and more dependent on dense untangling maps.

2. Decomposing Merge Accuracy

Final merge accuracy conflates alignment quality with several distinct costs, which we isolate. We compare two ways of applying the *same* alignment maps: an idealised representation-space merge and the corresponding practical weight-space merge. Representation-merge asks what merging would achieve if the alignment maps could be applied directly to activations at their intended boundaries; weights-merge asks how much of that idealised merge accuracy survives once the same maps are folded into parameters and the models averaged.

2.1. Layerwise Latent Alignment

We consider N networks of identical architecture trained from different random seeds, focusing on $N = 2$. Despite having identical architectures, each network learns its own private internal representations, making activations from different models incompatible for direct merging. We therefore introduce a shared latent interface: for each model n, at each layer l (for residual architectures, including residual-block boundaries), an encoder E_n^l maps activations from the model's private representation into a shared space, and a decoder D_n^l maps them back. For convolutional networks, E_n^l and D_n^l are bias-free 1×1 convolutions, with the nonlinearity applied in the shared space.

The framework is agnostic to how (E, D) are obtained. We instantiate the maps using methods from three families: permutation and many-to-many assignment (Ainsworth et al., 2023; Stoica et al., 2024; Singh & Jaggi, 2020; Peña et al., 2023), CCA-based projections (Horoi et al., 2024) (including the CCA-AD refinement Section E), and trained linear encoder–decoder maps optimised by cross-prediction reconstruction between the two models. The trained maps draw on model-stitching and activation-matching precedents (Lenc & Vedaldi, 2015; Bansal et al., 2021; Csiszárik et al., 2021) and are closely related to Foldable SuperNets (Kinderman et al., 2025); we use them only as diagnostic references for stronger dense alignments, with configurations in Appendix F.

2.2. Two Merge Procedures

Both procedures share the same alignment maps; they differ only in whether the maps are applied explicitly at runtime or committed to parameters before averaging.

Representation-merge. The same input x is passed through both models in parallel. At each selected layer l, activations are encoded into the shared space, averaged there, and decoded back into each model's private space before computation continues. Each model keeps its own BatchNorm (BN) layers, so representation-merge preserves the normalisation behaviour seen during training. Final logits are averaged at the classifier.

Weights-merge. The encoders and decoders are folded algebraically into the surrounding conv/BN/linear layers, such that both models' layers output to the same shared latent space. The folded parameters are then averaged layerwise into a single merged model, whose BN running statistics are recalibrated afterwards.

When each encoder/decoder folds into an affine map, representation-space averaging and parameter-space averaging are functionally equivalent (see Appendix C). Whether this holds for the convolutional models considered here is determined by the handling of BN, so we consider two strategies:

BN-preserved: align conv and BN, merge, recalibrate. BN affine parameters are transformed by the alignment maps and running statistics are recalibrated post-merge. This is exact for scaled permutations and a channelwise approximation for dense maps.

BN-absorbed: absorb BN into conv weights, align, merge. BN normalisation and affine parameters are fully folded into the conv weights, then encoders folded on top before merging. Exact for arbitrary linear maps but removes the live normalisation handle and collapses most weights-merged models to near chance (Jordan et al., 2023; Qu & Horvath, 2024) (Appendix D).

2.3. Decomposing the Representation–Weights Gap

For sequential architectures such as VGG-16-BN the comparison is direct: representation-merge applies the maps at each layer l, and weights-merge folds the same maps into the adjacent parameters. Residual architectures need a finer comparison, because one residual-stream alignment may be folded into several writers and readers of the same stream: block output convolutions, subsequent block input convolutions, and shortcuts.

Figure 1 illustrates the resulting staircase. The first three steps are variants of representation-merge evaluated at pro-

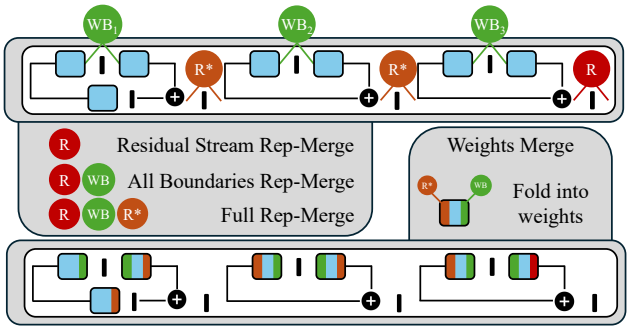

Figure 1. Staircase decomposition for a residual stage. The diagram shows how the alignment maps are applied to each of the models to be merged. At each selected layer the two models' representations are averaged in the shared latent space. *Top (representation-merge):* the maps are applied at runtime—residual-stream rep-merge uses only the stage-output alignment R (red), all-boundaries adds the within-block boundaries *WB* (green), and full rep-merge additionally applies R at every within-stage residual position (R^*, orange), matching the structure weights-merge imposes. *Bottom (weights-merge):* all of $\{R, WB, R^*\}$ are folded into the surrounding convolutions before weights merging; folding is shown as the map colours blended into each weight tensor.

gressively larger sets of alignment boundaries, and the last is weights-merge, so the staircase traces a single path from the idealised merge to the practical one:

- **Residual-stream rep-merge**: average codes only at residual stage outputs, treating each residual stage as a black box and merging only at stage outputs.
- **All-boundaries rep-merge**: additionally average codes at the within-block alignment boundaries.
- **Full rep-merge**: additionally apply the residual-stream alignment at every block boundary within the stage, precisely as weights-merge requires.
- **Weights-merge**: fold the same maps into parameters and merge the folded models.

Each transition defines a gap with its own interpretation.

Base accuracy → residual-stream gap. This is the cost of merging in idealised conditions: residual stages are treated as black boxes and merged only at their outputs.

Residual-stream → all-boundaries gap. This is the additional cost of requiring compatibility inside a residual stage, not only at its output. Two stages can produce compatible outputs while distributing computation differently across their constituent blocks. As a result, they may merge successfully at the stage output yet remain misaligned at corresponding internal positions.

All-boundaries → full gap. This is the architectural constraint imposed by the residual stream: because the stream is read and written at many positions, an architecture-preserving merge must reuse a single residual-stream alignment at every position it touches, not only the one where it was fitted—a constraint of architecture preservation rather than of any particular merge method.

Full → weights-merge gap. This gap captures the remaining differences between representation- and weights-merge under BN-preserved folding. First, BN-preserved folding is exact only for scaled permutations (§2.2); dense alignments (CCA and Train) therefore incur approximation error when folded into BN. Second, full rep-merge runs each model's conv–BN stack independently before averaging, including during BN recalibration. In contrast, weights-merge averages the folded parameters before normalisation, leading to slightly different recalibrated BatchNorm statistics.

Importantly, each gap is a marginal-cost estimate rather than a clean isolation: every more-constrained variant introduces more encoder–decoder pairs and hence more merge points, so losses compound down the staircase regardless of alignment quality. The compounding is lightest for permutations, whose exact reconstruction ($D \circ E = I$) and ReLU-commutation make each gap free of reconstruction and nonlinearity artefacts; for dense maps the gaps are upper bounds on the costs they are meant to isolate.

3. Findings

We evaluate on CIFAR-100 using VGG-16-BN and ResNet-20 across width multipliers $\{0.25, 0.5, 1.0\}$ for VGG and $\{1, 4, 8\}$ for ResNet. Table 1 reports representation-merge variants and weights-merge accuracy across alignment methods under BN-preserved folding. For ResNet, the representation-merge variants form the staircase introduced in §2.3; for VGG, the staircase collapses to a single full representation-merge metric. BN ablations are reported in Appendix D. All results are mean ± std over three seeds. Base accuracies are reported in Appendix A.[1]

3.1. Architecture preservation, not folding, is the dominant post-alignment cost

The staircase separates costs intrinsic to alignment quality from structural and folding costs (§2.3), making each independently addressable.

At ResNet $w = 1$ with Permute, the staircase reads $52.2\% \to 28.4\% \to 14.5\% \to 13.1\%$, illustrating the pattern: the all-boundaries → full gap (13.9pp) substantially exceeds the full → weights gap (1.4pp), while residual-stream rep-merge is already reasonably high. The full →

[1] Our CCA numbers differ from those in Horoi et al. (2024); we attribute this to differences in pretrained checkpoints. We verified our CCA implementation against the original codebase, which reproduces our scores on our checkpoints.

Table 1. Representation-merge and weights-merge accuracy (%) under BN-preserved folding. ResNet rows report the staircase from §2.3; VGG reports full rep-merge only. Mean ± std over 3 seeds; best in row **bold**. Trained maps use CCA-AD initialisation. BN-absorbed comparisons are in Appendix D.3.

Model	×	CCA	CCA-AD	Perm	ZipIt	Train
Residual stream rep-merge						
RN	1	49.1±0.6	55.1±0.5	52.2±1.2	41.0±0.9	**60.6**±0.3
RN	4	70.8±0.3	72.8±0.5	73.9±0.3	70.7±0.4	**74.5**±0.2
RN	8	75.7±0.1	76.6±0.2	**77.8**±0.1	76.5±0.2	77.3±0.3
All boundaries rep-merge						
RN	1	41.2±1.3	44.6±1.2	28.4±0.9	25.5±1.6	**54.6**±0.5
RN	4	68.7±0.5	69.8±0.4	62.4±0.6	60.7±0.5	**73.0**±0.2
RN	8	74.7±0.3	75.2±0.4	71.0±0.4	70.8±0.2	**75.6**±0.4
Full rep-merge						
RN	1	21.6±2.8	22.7±1.8	14.5±1.9	12.3±1.1	**31.2**±1.6
RN	4	54.3±0.4	**55.7**±0.3	43.6±0.5	41.3±0.7	54.2±0.6
RN	8	65.1±0.5	**66.7**±0.5	55.5±1.8	55.2±0.6	57.1±1.1
VGG	0.25	35.2±2.3	39.9±1.8	22.0±1.4	20.1±0.9	**57.9**±0.2
VGG	0.5	57.4±0.2	56.7±0.3	40.1±1.0	40.0±0.7	**63.6**±0.4
VGG	1	65.6±0.3	65.0±0.1	55.8±0.4	54.7±0.3	**65.7**±0.0
Weights-merge						
RN	1	9.2±1.0	10.4±1.8	13.1±0.5	10.4±0.5	**25.5**±2.2
RN	4	49.8±1.7	52.6±1.0	44.4±0.4	41.1±0.2	**53.9**±0.4
RN	8	64.3±0.7	**65.7**±0.2	57.8±0.9	57.5±0.7	59.6±0.2
VGG	0.25	21.6±1.2	25.6±0.4	24.2±1.0	22.2±1.5	**57.9**±0.2
VGG	0.5	54.5±0.5	53.0±0.5	42.2±0.8	42.6±1.0	**60.6**±0.4
VGG	1	**64.9**±0.3	63.6±0.1	56.9±0.6	55.8±0.3	64.5±0.2

weights gap is small across most settings; the one systematic exception is dense maps at narrow width under BN-preserved folding (CCA at $w = 1$: $21.6\% \to 9.2\%$), where folding a dense, channel-mixing alignment into the BatchNorm is only approximate (see Appendix D). Even there, the post-alignment architectural constraint is the larger factor. Figure 2 and Table 1 give the full results.

The implication is that improving alignment maps—the target of most recent merge-method development—cannot in general eliminate the all-boundaries → full gap: this gap exists even when the fitted-boundary alignment is good, because it arises from the requirement to reuse that alignment at positions where it was not fitted (Appendix B). The trained maps make this concrete: even with the strongest alignment we obtain, ResNet $w = 1$ still loses 23.4pp across this transition (all-boundaries $54.6\% \to$ full 31.2%).

VGG as sanity check. For VGG, the sequential architecture collapses the staircase to a single rep-merge metric, merging representations between every conv–BN stack (*full*).

The full → weights gap is small for permutations and trained maps and grows mainly for narrow CCA-style dense maps,

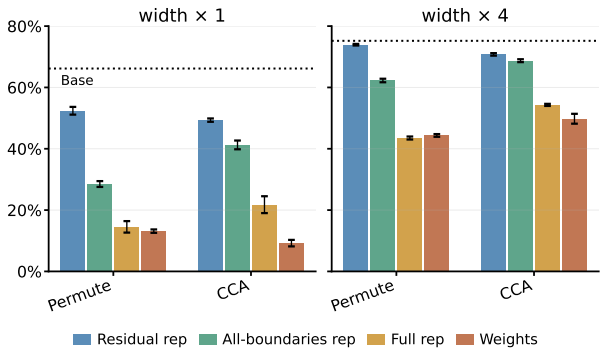

Figure 2. Staircase decomposition on ResNet-20 at two representative widths ($w = 1$ and $w = 4$). Within each panel we compare Permute and CCA; the bars are the progressively more constrained variants — residual-stream rep-merge, all-boundaries rep-merge, full rep-merge, and weights-merge. The dotted Base line is the average single-model accuracy (Table 2), the performance a successful merge aims to recover. Error bars are ±1 std over three seeds. Full results, including the saturated $w = 8$ setting, are in Table 1.

again reflecting the BN-preserved approximation.

Predicted intervention. If reusing one alignment across multiple residual positions drives the all-boundaries → full gap, then adding identity-initialised channel maps in skip connections—letting different block boundaries within a stage adopt independent alignments—should reduce it, at the cost of exact architecture preservation. This is a falsifiable architectural test that the framework suggests but we do not evaluate here.

3.2. Closed-form methods underperform in narrow models

Having isolated the architectural constraint, we now ask whether alignment quality itself is a limiting factor. We compare methods at the staircase step that best isolates this: all-boundaries rep-merge for ResNet and full rep-merge for VGG (where the staircase collapses to a single metric).

At wide width, the trained-map advantage largely closes. On ResNet $w = 8$, all-boundaries rep-merge reaches 75.6% for trained maps, compared with 75.2% for CCA-AD and 74.7% for CCA; on VGG $w = 1$, trained maps reach 65.7%, essentially matching CCA at 65.6%. This is consistent with the single-basin conjecture (Entezari et al., 2022): as width grows, closed-form alignments become increasingly sufficient.

At narrow width, trained maps exceed every closed-form method. The gap is 18pp at VGG $w = 0.25$ (trained 57.9% vs CCA-AD 39.9%) and 10pp at ResNet $w = 1$ (trained 54.6% vs CCA-AD 44.6%), shrinking to near zero at wide settings across both architectures.

A partial mechanism. Training optimises $D \circ \mathrm{ReLU} \circ E \approx I$ directly, the composition that executes at each aligned layer. CCA-based methods are not designed for this through-nonlinearity reconstruction objective; CCA-AD partially addresses the decoder side (Appendix E), but a significant gap remains. This explains why trained maps outperform closed-form methods, though it does not fully account for why the gap is concentrated at narrow width—we conjecture this reflects more entangled representations at narrow width that closed-form methods cannot disentangle.

3.3. Conclusion

Weights-merge accuracy is a single number that conflates the quality of an alignment with the costs of realising it inside an architecture, which is why a poor merge is so readily read as a poor alignment. Our central contribution is a way to take that number apart: a staircase of operating points that begins at an idealised representation-merge (the alignment applied directly to activations at runtime) and, one constraint at a time, walks toward the practical weights-merge it is meant to approximate. Each transition attributes the accuracy lost across it to an identifiable contributor: the alignment itself, the requirement that one alignment stay compatible at every position the architecture forces it to serve, or the algebra of folding and normalisation. Although the decomposition does not always perfectly isolate each effect, it turns a single failure number into a profile of more interpretable parts, exposing the design and implementation choices behind a merge process.

Applied to CIFAR-100 with VGG-16-BN and ResNet-20, the staircase shows that narrow-ResNet failure is driven not by alignment quality or by the final fold but by the cost of preserving the architecture — one residual-stream alignment forced to serve many positions — while also showing that, once that cost is set aside, closed-form alignment quality is itself still limiting at narrow width, where trained maps substantially outperform it. Neither conclusion is reachable from weights-merge accuracy alone. The construction assumes nothing specific to these settings: it requires only that alignment maps can be inserted at boundaries and either run at runtime or folded into parameters, which holds for any alignment method and any architecture with identifiable merge boundaries. We therefore offer the staircase less as a result about two networks than as a diagnostic that merging researchers can reuse: to ask, of any new method, task, or architecture, not merely whether a merge succeeds but which costs along the way it actually pays, and thus whether the path to better merging lies in stronger alignment or in what follows once an alignment is fixed.

References

Ainsworth, S., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Akash, A. K., Li, S., and Trillos, N. G. Wasserstein barycenter-based model fusion and linear mode connectivity of neural networks. *arXiv preprint arXiv:2210.06671*, 2022.

Bansal, Y., Nakkiran, P., and Barak, B. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

Boufalis, O., Carrasco-Pollo, J., Rosenthal, J., Terres-Caballero, E., and García-Castellanos, A. Symmetry-aware graph metanetwork autoencoders: Model merging through parameter canonicalization. *arXiv preprint arXiv:2511.12601*, 2025.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Csiszárik, A., Kőrösi-Szabó, P., Matszangosz, A., Papp, G., and Varga, D. Similarity and matching of neural network representations. *Advances in Neural Information Processing Systems*, 34:5656–5668, 2021.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

Horoi, S., Orozco Camacho, A. M., Belilovsky, E., and Wolf, G. Harmony in diversity: Merging neural networks with canonical correlation analysis. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver,

N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18815–18832. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/horoi24a.html.

Ito, A., Yamada, M., Chijiwa, D., and Kumagai, A. Do we really need permutations? impact of width expansion on linear mode connectivity. *arXiv preprint arXiv:2510.08023*, 2025a.

Ito, A., Yamada, M., and Kumagai, A. Analysis of linear mode connectivity via permutation-based weight matching: With insights into other permutation search methods. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=lYRkGZZi9D.

Jordan, K., Sedghi, H., Saukh, O., Entezari, R., and Neyshabur, B. REPAIR: REnormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gU5sJ6ZggcX.

Kinderman, E., Hubara, I., Maron, H., and Soudry, D. Foldable supernets: Scalable merging of transformers with different initializations and tasks. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=6FqwLestHv.

Lenc, K. and Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

Nasery, A., Hayase, J., Koh, P. W., and Oh, S. Pleas-merging models with permutations and least squares. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 30493–30502, 2025.

Peña, F. A. G., Medeiros, H. R., Dubail, T., Aminbeidokhti, M., Granger, E., and Pedersoli, M. Re-basin via implicit sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20237–20246, 2023.

Qu, X. and Horvath, S. Vanishing feature: Diagnosing model merging and beyond. *arXiv preprint arXiv:2402.05966*, 2024.

Sharma, E., Roy, D. M., and Dziugaite, G. K. The non-local model merging problem: Permutation symmetries and variance collapse. *arXiv preprint arXiv:2410.12766*, 2024.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

Stoica, G., Bolya, D., Bjorner, J. B., Ramesh, P., Hearn, T., and Hoffman, J. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LEYUkvdUhq.

Thasarathan, H., Forsyth, J., Fel, T., Kowal, M., and Derpanis, K. G. Universal sparse autoencoders: Interpretable cross-model concept alignment. In *Forty-second International Conference on Machine Learning*, 2025.

Table 2. Single-model test accuracy (%) on CIFAR-100. Mean ± std over 3 seeds.

Model	×	Accuracy
RN	1	66.2±0.5
RN	4	75.2±0.3
RN	8	77.6±0.1
VGG	0.25	61.1±0.2
VGG	0.5	67.2±0.6
VGG	1	71.4±0.6

A. Single-model baselines

Table 2 reports single-model test accuracy for each architecture and width—the Base reference a successful merge aims to recover (the dotted line in Figure 2).

B. Residual-stage folding and architecture preservation

Figure 3 illustrates where the trained encoder–decoder maps are fit and where they are folded into the network's weights. This distinction is central to the residual-network gap discussed in §3.1.

In sequential architectures such as VGG, each alignment boundary corresponds to a local segment of computation: a decoder feeds the next segment and an encoder returns its output to the shared space. Folding therefore inserts the same maps into the immediately adjacent layers. When the maps are affine-foldable, the representation-space and weight-space constructions are local realisations of the same computation.

Residual stages break this locality. A stage output is written by several paths: the residual branch of each block and, in downsampling blocks, a shortcut convolution. Conversely, the next stage reads the residual stream at several sites. After folding, a single residual-stream alignment must therefore be absorbed into multiple geographically distinct convolutions. These fold targets sit at different depths in the stage and see different activation statistics from the boundary where the alignment was fit.

This structural asymmetry—the cost of architecture preservation discussed in §3.1—is what the representation–weights gap measures. Representation-merge applies each encoder and decoder at its fitted boundary. Weights-merge instead asks the same boundary map to serve simultaneously as a transform on several writers and readers. Large gaps on narrow ResNets therefore need not imply that the boundary alignment is poor; they can arise because the architecture lacks a local parameterisation capable of realising that alignment after folding.

The within-block trained maps shown in Figure 3 do not have the same multi-site folding problem. They are fit and folded within a single residual block, so their representation–weights gap is correspondingly smaller. The large gaps in §3.1 are driven by the residual-stream alignment, where one boundary-level map is reused across an entire stage.

C. Affine-folding equivalence

Consider the computation between two consecutive alignment boundaries, ℓ and $\ell+1$. For model i, let $F_{i,\ell}$ denote this segment, let $D_{i,\ell}$ map from the shared representation at boundary ℓ into model i's private space, and let $E_{i,\ell+1}$ map back into the shared space at boundary $\ell+1$. The aligned segment is

$$G_{i,\ell} = E_{i,\ell+1} \circ F_{i,\ell} \circ D_{i,\ell}.$$

If this folded segment is affine, $G_{i,\ell}(x) = M_{i,\ell}x + c_{i,\ell}$, then averaging aligned representations is equivalent to averaging the folded affine parameters:

$$\frac{1}{2}\left(G_{A,\ell}(x) + G_{B,\ell}(x)\right) = \frac{1}{2}(M_{A,\ell}+M_{B,\ell})x + \frac{1}{2}(c_{A,\ell}+c_{B,\ell}).$$

The extension to N models replaces the two-model average with an average over N. Thus, when each aligned segment can be realised as a local affine map, representation-merge and weights-merge compute the same function segmentwise. Their disagreement measures violations of this locality or affine-foldability assumption, or post-merge normalisation effects.

D. BatchNorm folding details and ablations

D.1. BN-absorbed and BN-preserved folding

The two BN folding modes used in the paper expose a tradeoff between exact affine folding and post-merge recalibration. Figure 4 illustrates the distinction.

Under **BN-absorbed** folding, BN is first absorbed into the preceding convolution using its running mean, running variance, scale, and shift parameters. The alignment map is then folded into this affine layer. This gives an exact per-model affine fold for arbitrary linear encoders, but removes the live BN layer from the merged computation. After parameter averaging, there is no normalisation module at that location whose statistics can be recalibrated.

Under **BN-preserved** folding, only the encoder is folded into the preceding convolution ($\mathrm{conv}^* = E\,\mathrm{conv}$), and the channelwise BatchNorm is kept live with its affine scale reset to $\mathrm{diag}(E\gamma)$. The merged model therefore retains a recalibratable normalisation layer. This is exact only when E is a scaled permutation: for a dense E the exact transform of the BN affine, $E\,\mathrm{diag}(\gamma)$, is itself dense, and replacing it with the channelwise $\mathrm{diag}(E\gamma)$ drops the cross-channel coupling E would introduce. The fold is therefore approximate

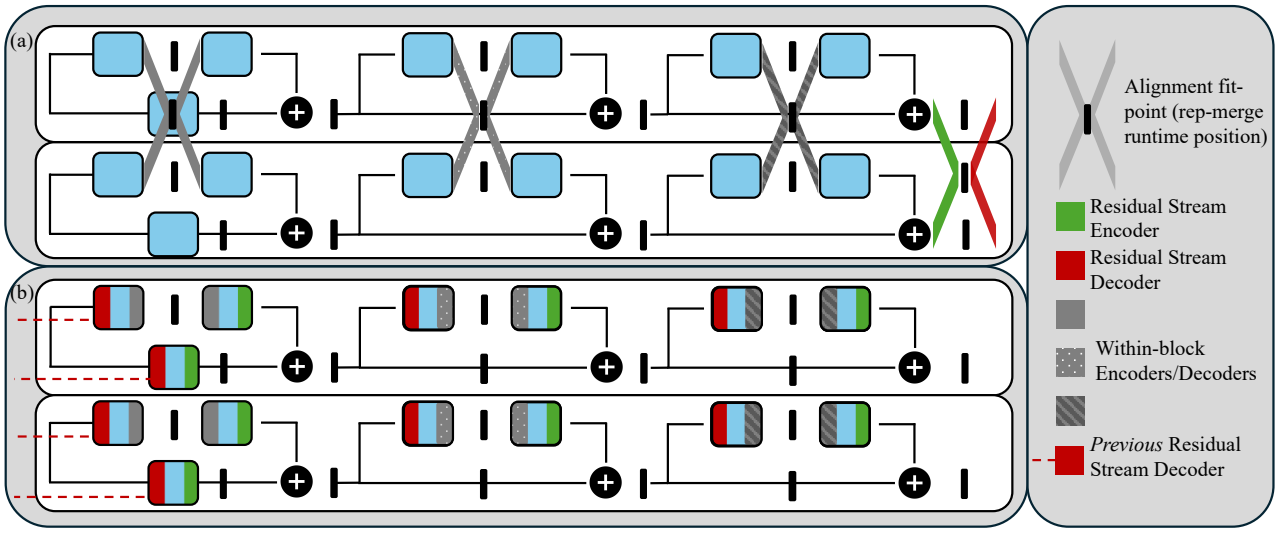

Figure 3. Architecture diagram for one ResNet stage. (a) Pre-fold view: encoder/decoder pairs are fit at specific activation boundaries, where paired activations from the two models can be sampled and aligned. Representation-merge applies the alignment maps at these same boundaries during the forward pass. (b) Post-fold view: weights-merge folds the same maps into surrounding weights. A residual-stream encoder fit at the end of a stage must be folded into multiple writers to that stream, including block output convolutions and downsample shortcuts; the previous stage's decoder must similarly be folded into multiple readers of the stream. Thus a single boundary alignment is fit at one statistical location but must act at several parameter locations after folding.

for dense alignments, with the post-merge recalibration of the running statistics partially compensating. Empirically this approximation is preferable to removing the live normalisation handle entirely, as shown in Table 3.

D.2. Universal BatchNorm

Universal BatchNorm (UBN) is an alternative variant in which a shared BN module is inserted between the encoder and the shared latent space at each boundary and observes codes from both models during trained-map optimisation. UBN preserves a live normalisation layer after merging while retaining the exact affine fold of BN-absorbed mode. It is therefore a natural test of whether the weakness of BN-absorbed folding is simply the absence of any live normalisation downstream of the merge.

Empirically, UBN is not sufficient to match the best BN-preserved results: only one of the trained-map configurations selected for Table 1 uses it (see Appendix F), and Table 3 reports BN-absorbed results with UBN throughout for completeness. We interpret this as evidence that the *location* of the live normalisation matters. UBN normalises only after the absorbed convolution and the encoder have been applied, whereas BN-preserved folding keeps the original BN between the merged convolution and the encoder, where merged pre-encoder features can be recalibrated. Once the original BN statistics have been baked into the averaged convolution weights, a downstream UBN cannot undo all of the resulting mismatch.

D.3. Additional BatchNorm-mode results

Table 3 reports the full BN-mode comparison across method, architecture, and width. These results support the BN discussion in §2: exact affine folding under BN-absorbed mode is not by itself sufficient for successful merging.

Under BN-absorbed folding, Permute and ZipIt collapse to near chance across architectures and widths, despite the per-model fold being algebraically cleaner. By contrast, BN-preserved folding retains live normalisation and yields non-trivial weights-merge performance across methods. CCA is the main exception under BN-absorbed folding, consistent with the interpretation that whitening provides a substitute for the lost re-normalisation ability. Qu & Horvath (2024) further decompose this failure mode, relating REPAIR-style normalisation to rescaling and reshifting collapsed features.

E. CCA-AD initialisation for trained maps

Trained-map results in Table 1 are initialised from a refinement of CCA-Merge (Horoi et al., 2024), which we call CCA with analytic decoder, or CCA-AD. This initialisation is especially helpful for narrow models, where the alignment problem is hardest.

Standard CCA-Merge fits an encoder on post-ReLU activations and uses a pseudoinverse decoder. This creates a domain mismatch: the encoder is selected using post-ReLU statistics, while the decoder must support the through-ReLU computation used by the next layer. CCA-AD keeps the CCA encoder but replaces the pseudoinverse decoder with

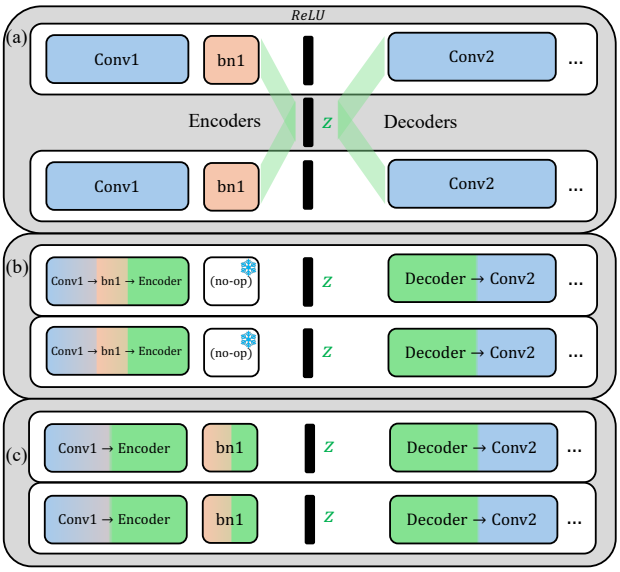

Figure 4. Trained map structure and BatchNorm folding modes. (a) The trained encoder–decoder primitive: linear encoders and decoders map private representations into and out of a shared latent space. (b) BN-absorbed folding: BN is absorbed into the preceding convolution using fixed statistics, giving an exact affine layer but leaving no live normalisation at the merge point. (c) BN-preserved folding: the encoder is folded into the convolution while BN remains live and recalibratable. This preserves a normalisation handle after merging, but is exact only for scaled-permutation alignments.

the analytic least-squares solution for reconstruction through the shared ReLU.

For encoder E, CCA-AD solves

$$D^* = \arg\min_D \mathbb{E}\left[\|D\,\mathrm{ReLU}(Ex) - \mathrm{ReLU}(x)\|_2^2\right].$$

This is a linear least-squares problem in D. In practice, the solution is computed from sampled activations at the corresponding boundary. The encoder is unchanged from CCA; only the decoder is replaced.

CCA-AD appears in Table 1 both as a closed-form baseline in its own right and as the initialisation for the trained maps; as an incremental adjustment to CCA merge, we do not propose it as a novel merging method. Its role is to provide a stronger starting point in narrow-width regimes, where ordinary CCA is less effective. This is important for interpreting the trained-map row in Table 1: the row demonstrates that substantially better alignments are reachable, but it combines optimisation with a strong CCA-AD initialisation.

Table 3. Alignment and weights-merge accuracy (%) across architectures, widths, and methods. Top block: representation-merge accuracy (pre-fold, every-boundary), a pure alignment-quality metric. Bottom block: weights-merge accuracy after folding. Mean ± std over model pairs; best in row **bold**. `BN-absorbed` and UBN throughout.

Model	×	CCA	CCA-AD	Perm	ZipIt	Train
Representation-merge (pre-fold)						
VGG	0.25	9.4±1.4	15.6±2.3	1.3±0.0	1.2±0.2	**57.9**±0.2
VGG	0.5	45.3±0.3	37.6±1.8	1.0±0.0	1.4±0.4	**63.6**±0.4
VGG	1	61.9±0.9	52.5±0.4	1.1±0.1	1.1±0.1	**65.7**±0.0
RN	1	5.0±2.8	29.3±5.7	28.1±6.1	19.0±0.9	**60.6**±0.3
RN	4	41.0±4.9	68.4±0.2	54.9±1.5	54.1±2.0	**74.5**±0.2
RN	8	64.3±3.1	75.4±0.4	60.0±0.3	61.1±1.5	**77.3**±0.3
Weights-merge (post-fold)						
VGG	0.25	9.4±1.4	15.6±2.3	1.3±0.0	1.2±0.2	**57.9**±0.2
VGG	0.5	45.3±0.3	37.7±1.8	1.0±0.0	1.4±0.4	**60.6**±0.4
VGG	1	61.9±0.9	52.5±0.4	1.1±0.1	1.1±0.1	**64.5**±0.2
RN	1	8.8±1.8	9.4±1.6	5.1±1.8	4.8±0.4	**25.6**±2.2
RN	4	20.0±2.7	18.8±1.9	7.5±0.5	5.9±0.3	**53.8**±0.1
RN	8	25.8±0.1	24.0±0.9	7.7±0.4	7.9±0.6	**59.1**±1.1

F. Trained encoder–decoder map configurations

Table 4 reports the per-cell configuration of the best trained encoder–decoder map result used in Table 1. These configurations are included for reproducibility. We treat these trained maps as an existence proof that better alignments are achievable, rather than as a proposed standalone merging method.

Our training objective follows a cross-prediction setup between paired model activations, similar in role to the foldable maps of Foldable SuperNets (Kinderman et al., 2025). The training regime was inspired by Universal Sparse Autoencoders (Thasarathan et al., 2025), but the resulting modules are simply linear encoder–decoder alignment maps used here as diagnostic references.

Table 4. Best-performing trained U-AE configuration for each architecture/width. Reported are the selected hyperparameters and resulting accuracies. Repr = representation-merge (pre-fold), Merge = weights-merge (post-fold). Mean ± std over seeds. All configurations initialise via CCA-AD, with additional initialisation noise applied via linear interpolation between a Kaiming random init and the CCA-AD tensors.

Model	×	BN Fuse	Ep	FT	Repr	Merge
VGG	0.25	full	0	100	57.9±0.2	57.9±0.2
VGG	0.5	none	10	20	63.6±0.4	60.6±0.4
VGG	1	none	10	0	65.7±0.0	64.5±0.2
RN	1	none	10	20	60.6±0.3	25.6±2.2
RN	4	none	0	20	74.5±0.2	53.8±0.1
RN	8	none	0	20	77.3±0.3	59.1±1.1