IS DEPTH HETEROGENEITY A BARRIER TO MODEL MERGING?

Nour Shaheen[1,2,3] **Sarath Chandar**[1,2,3,6] **Boris Knyazev**[2,4,5*] **Ekaterina Lobacheva**[1,2,4*]
[1]Chandar Research Lab [2]Mila – Quebec AI Institute [3]Polytechnique Montréal
[4]Université de Montréal [5]Samsung AI Lab, Montreal [6]Canada CIFAR AI Chair
Correspondence: nour.shaheen@mila.quebec

ABSTRACT

Model merging offers a way to combine the capabilities of several networks at test time without retraining or additional finetuning, but most merging methods assume identical architectures. Depth differences are commonly viewed as a major obstacle because they remove clear layer correspondences. We test this assumption by merging residual networks that differ only in depth, using a simple training-free pipeline based on identity expansion and permutation alignment. Across both same-task and multitask image classification experiments, heterogeneous merges closely match homogeneous ones. The results suggest that, for residual networks, depth mismatch is not the main barrier to effective model merging, and that the main difficulty in model merging comes from aligning independently trained weights in a homogeneous setting.

1 INTRODUCTION

Modern deep learning has produced a large number of pretrained and finetuned models, each specializing in different data distributions or tasks. Combining or improving performance at test-time by retraining or joint finetuning of these models is often impractical, which motivates **model merging** (see Appendix A for an overview of related works). Most merging methods are designed for homogeneous settings, where models share the same architecture and layer structure, and often similar initializations. Under these assumptions, simple operations such as weight averaging or linear interpolation can already perform well (Izmailov et al., 2018; Wortsman et al., 2022). When models are trained independently and thus don't share the same optimization trajectory, merging becomes more complicated and requires matching the models to take into account symmetries such as neuron permutations (Ainsworth et al., 2023). Further, if networks have different architectures, mismatch between their structures has to be reconciled as well.

Architectural heterogeneity can arise at many levels. Differences in width are generally manageable since layer correspondences remain clear and neuron correspondences must be resolved regardless due to independent training. Theus et al. (2025) and Xu et al. (2024) successfully demonstrate that standard homogeneous merging methods can be easily expanded for this case. Depth heterogeneity, however, removes explicit layer correspondences and makes fusion ambiguous. Xu et al. (2024); Nguyen et al. (2023) propose novel approaches for matching or "mapping" layers between models of differing depths, before expanding the model and then merging. Although these methods show positive results in small scale experiments, it is not obvious if they are a viable solution for larger-scale tasks or more recent merging methods. Moreover, it remains unclear how much additional difficulty depth differences introduce, if at all, since naïve solutions are underexplored.

Our work isolates depth heterogeneity. We investigate whether merging models with different depths is inherently more difficult than merging models of equal depth under two regimes: a same-task setting and a multitask setting. Surprisingly, a simple heterogeneous baseline with naïve expansion and permutation alignment, without sophisticated layer mapping, achieves performance comparable to homogeneous merging across a wide range of model sizes and tasks. For residual networks, depth heterogeneity does not appear to be the dominant obstacle. Progress in heterogeneous merging

*Equal senior authorship.

therefore, at this stage, depends more on improving methods for independently initialized models than on increasingly complex cross-architecture alignment strategies. With continual progress in the former, the latter, however, may become more relevant in the future.

2 EXPERIMENTAL SETUP

Architectures and Setting All experiments are conducted on convolutional neural networks from the **ResNet** family (He et al., 2015). Our setting is *depth heterogeneous only*: all models share the same architecture design and channel widths within each stage, and differ exclusively in the number of residual blocks. For the same-task ImageNet experiments, we evaluate merges among ResNet17, 29, 35, 50, 101, and 152 variants, which we define in Appendix B in Table 1. For simplicity, for the multitask pairs, we limit our study to standard deeper backbones (ResNet50, 101, and 152).

Tasks and Datasets We evaluate both *same-task* and *multitask* merging regimes. In the first regime, independently trained ResNets are merged pairwise and evaluated on the **ImageNet** classification benchmark (Deng et al., 2009). For multitask experiments, all models are first pre-trained on ImageNet and then finetuned separately on the following downstream datasets: **Describable Textures Dataset (DTD)** (Cimpoi et al., 2013), **CUB-200-2011** (Wah et al., 2011), **NABirds** (Van Horn et al., 2015), **Street View House Numbers (SVHN)** (Netzer et al., 2011), **MNIST** (LeCun et al., 1998), **EuroSAT** (Helber et al., 2019), and **DomainNet-Infograph** (Peng et al., 2019). We merge only the following task pairs, evaluating each pair in both directions: DTD–CUB, SVHN–MNIST, SVHN–CUB, MNIST–EuroSAT, CUB–Infograph, and NABirds–CUB. Models within each pair are finetuned from *different* initializations across multiple seeds.

Model Expansion We perform depth expansion to embed shallower networks into the topology of deeper ones using a Net2Net–style method (Chen et al., 2016). Residual stages are matched one to one, and extra residual blocks are inserted as identity mappings. This preserves the original function of the shallower model in a deeper architecture. For each residual stage $s \in \{2, 3, 4, 5\}$ with n_s blocks and target depth N_s, we keep blocks $B_{s,1}, \ldots, B_{s,n_s}$ unchanged and append identity blocks $I_{s,n_s+1}, \ldots, I_{s,N_s}$. We perform no layer matching, which makes our approach a significantly simplified version of Xu et al. (2024); Nguyen et al. (2023). Details are in Appendix C.

Permutation Alignment and Merging We then apply permutation alignment following the *Git Re–Basin* paradigm (Ainsworth et al., 2023). We perform activation–space channel alignment across corresponding layers, treating the deeper (unexpanded) model as a reference and permuting the neurons of the second model's layers. For the identity blocks added, permuting them essentially has no effect, and when they are merged with the deeper model's corresponding learned residual blocks, the residual output is effectively halved. For same-task models, all layers, including the classification head, are merged, whereas for multitask settings, merging is performed only up to (but excluding) the final task-specific classification heads. More specific details are provided in Appendix D.

Evaluation Protocol For same–task ImageNet experiments, merged models are evaluated directly on the ImageNet validation set. For multitask merges, we evaluate performance on each task separately using the classification head of the corresponding models: head A on task A and head B on task B, producing two task accuracies. Our main accuracy metric, the per-task average accuracy, is the average of both $\frac{1}{2}(\text{Acc}_m^A + \text{Acc}_m^B)$.

3 SAME-TASK SETTING

We study same-task (ImageNet–ImageNet) merging to isolate the effect of architectural heterogeneity from task mismatch. We merge each two independently trained ImageNet models, either homogeneously or heterogeneously, aggregate by sizes, and show results in Table 2 (Appendix E).

Homogeneous vs. Baseline Figure 1 (left) compares homogeneous merging performance against the original accuracies of the source models. While stronger base models generally have higher merged performance up to mid-scale architectures, we observe a drop afterwards for very large models, and attribute this to the growing number of parameters that must be reconciled during merging. We observe a clear "sweet spot" where improvements from higher base accuracy are balanced against increasing merging difficulty due to larger model sizes.

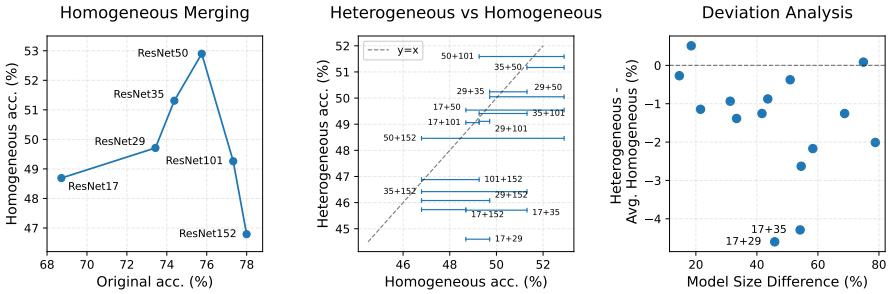

Figure 1: Same-task setting, varied model depths. (Left) Homogeneous merging vs. original non-merged performance. (Middle) Heterogeneous vs. homogeneous merging. (Right) Deviation analysis: difference between heterogeneous and average homogeneous merging vs. model size difference.

Heterogeneous vs. Homogeneous Figure 1 (middle) is a parity plot comparing heterogeneous and homogeneous merges. Heterogeneous merges are drawn as horizontal spans at their score on the y-axis, with the left and right endpoints marking the worst and best homogeneous accuracies on the x-axis. The plot exhibits a tight vertical accuracy band: heterogeneous results differ by only a few percentage points despite large architectural gaps, and most lie within or near the homogeneous range, frequently crossing the $y = x$ line, indicating they are not systematically worse. While moderate depth differences behave almost identically to homogeneous merges, there's a clear low-accuracy tail that includes pairs with very high depth mismatches (R17+R152, R29+R152, R35+R152).This may stem from the large number of inserted identity blocks increasing alignment ambiguity. We also observe reduced performance for shallow–shallow heterogeneous pairs (R17+R29, R17+R35) despite strong homogeneous results. We treat them as mild outliers and hypothesize that as pairs, they might have less representational redundancy to absorb permutation or expansion noise. Overall, deviations are small and mainly associated with extreme depth gaps or low model capacity.

Deviation Analysis Figure 1 (right) relates the heterogeneous–homogeneous accuracy difference to size mismatch, defined as the average per-stage normalized block count difference. For a pair of models, at each residual stage, we compute the absolute difference in their block counts, normalize by the block count of the larger model at that stage, and average across stages. We do this because total layer differences ignore how depth is distributed across residual stages, whereas a per-stage normalized measure better captures structural discrepancies between architectures. The plot shows a slight negative trend (Spearman correlation = -0.3893): similarly sized models yield heterogeneous performance close to or slightly above the homogeneous average, whereas larger architectural gaps tend to underperform. The shallow–shallow heterogeneous outliers are again visible. Importantly, the worst observed gap is only around -4 percentage points (much smaller than the 20–30 pp drop between homogeneous merges and their original counterparts before merging), showing that architectural size mismatch contributes only a small fraction of the overall merging difficulty relative to the intrinsic challenge of aligning independently trained weights.

4 Multitask Setting

To study the effects of task mismatch and difficulty on heterogeneous merging, we introduce two measures: **task similarity** (between two tasks) and **task simplicity**. We compute task similarity based on a Fréchet Inception Distance (FID)-style metric as in Heusel et al. (2018). Details on this computation and task similarities are in Appendix G. For task simplicity, we use as a proxy the average test accuracy of ResNet50 original models pretrained on ImageNet and fine-tuned on the task. Higher accuracy implies an easier task. For a pair of tasks, their combined simplicity is the average of the two. In the experiments, we combine ResNets 50, 101, 152 pairwise and report per-task average accuracy. All accuracies can be found in Appendix F.

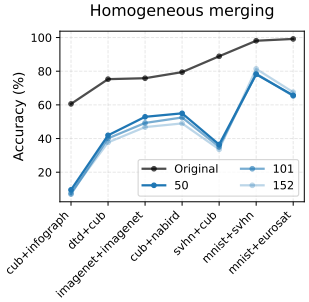

Figure 2: Homogeneous merging and original non-merged accuracy across task pairs.

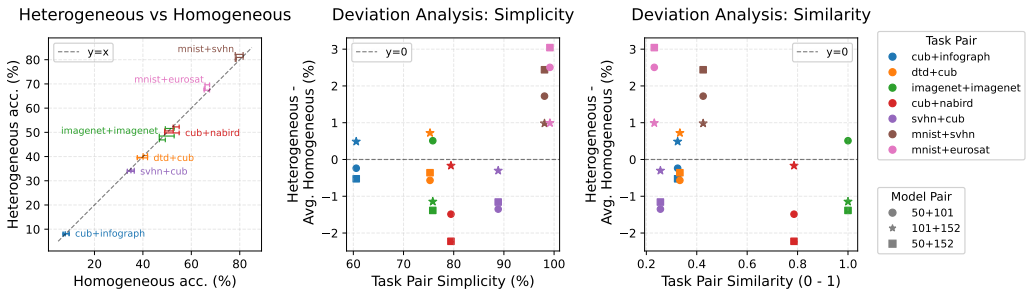

Figure 3: Multi-task setting, varied task pairs. (Left) Homogeneous vs. heterogeneous merging comparison. (Middle and right) Deviation analysis: difference between heterogeneous and average homogeneous merging vs. task pair simplicity and similarity respectively.

Homogeneous vs. Baseline In Figure 2, task pairs are ordered by increasing simplicity. We plot the average original ResNet50 accuracies alongside the homogeneous merging accuracies. We observe that homogeneous merging follows the same overall trend as the baselines but is consistently lower, with two pronounced drops for the most dissimilar task pairs (SVHN-CUB and MNIST-EuroSAT). Overall, homogeneous merging mostly preserves relative task performance but it has a consistent accuracy drop (due to the inherent difficulty of merging independently trained models), which increases for highly dissimilar tasks (SVHN-CUB and MNIST-EUROSAT, see Appendix G).

Heterogeneous vs. Homogeneous In Figure 3 (left), which mirrors the middle parity plot in the previous section, for each task pair, the heterogeneous merges are drawn as horizontal spans at their score on the y-axis, with the left and right endpoints marking the worst and best homogeneous accuracies on the x-axis. Because most horizontal spans cross the $y = x$ line, the heterogeneous scores are typically within the best–worst homogeneous range and frequently exceed the midpoint or upper bound. This indicates that, in multitask settings, heterogeneous scores closely follow homogeneous scores and the main factor influencing the merging accuracy of both of them is the task pair itself.

Deviation Analysis Figure 3 (middle and right) analyzes whether the deviation between heterogeneous and homogeneous scores is correlated with task simplicity or similarity. Overall, no strong monotonic relationship is observed with either of them. For simplicity, while low- to mid- simplicity pairs exhibit a very slight downwards trend, large positive deviations appear for the simplest task pairs. For similarity, while there may be an overall slight downwards trend, when we isolate low- to mid- similarity pairs (< 0.7), we find no consistent monotonic behavior. This suggests that heterogeneous merging gains are not reliably predicted by task simplicity or similarity, and are largely pair-specific. In both figures, deviations between heterogeneous and homogeneous accuracies remain within roughly [-2,2] percentage points. This confirms that the overall gap between the two is small, and that in the multitask regime as well, model heterogeneity contributes only a minor share of the total performance loss.

5 Conclusion

This work examined whether depth heterogeneity is truly an obstacle to merging models of different sizes. By isolating depth differences within a single architectural family and applying a simple training-free pipeline based on identity expansion and permutation alignment, we found that heterogeneous merges consistently matched the performance of homogeneous merges in both same-task and multitask settings. Performance drops appeared mainly with extreme depth gaps or very small models, and even then were minor compared to the gap between merged models and their independently trained baselines. Multitask results followed the same trend, with no consistent link to task similarity or simplicity. Overall, the primary challenge is aligning independently trained weights rather than architectural mismatch, suggesting future effort should focus more on improving core merging techniques than on complex cross-layer mappings. We plan to extend this analysis to transformer architectures and explore stronger merging algorithms, while carefully evaluating whether improvements observed in homogeneous merging transfer cleanly to heterogeneous settings or introduce trade-offs that degrade heterogeneous performance.

ACKNOWLEDGMENTS

We would like to thank Mila (mila.quebec) and its IDT team for providing and supporting the computing resources used in this work. Sarath Chandar is supported by the Canada CIFAR AI Chairs program, the Canada Research Chair in Lifelong Machine Learning, and the NSERC Discovery Grant. Ekaterina Lobacheva is supported by IVADO and the Canada First Research Excellence Fund.

REFERENCES

Sam Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models with shared underlying structure. *ICLR*, 2023.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer, 2016. URL https://arxiv.org/abs/1511.05641.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild, 2013. URL https://arxiv.org/abs/1311.3618.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

D. C. Dowson and B. V. Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. URL https://www.sciencedirect.com/science/article/pii/0047259X8290077X.

Felix Draxler, Ilya Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *ICML*, 2018.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitrii Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *NeurIPS*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification, 2019. URL https://arxiv.org/abs/1709.00029.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitrii Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *NeurIPS*, 2022.

Anshul Nasery, Jonathan Hayase, Pang Wei Koh, and Sewoong Oh. Pleas – merging models with permutations and least squares, 2025. URL https://arxiv.org/abs/2407.02447.

Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL https://api.semanticscholar.org/CorpusID:16852518.

Dang Nguyen, Trang Nguyen, Khai Nguyen, Dinh Phung, Hung Bui, and Nhat Ho. On cross-layer alignment for model fusion of heterogeneous neural networks, 2023. URL https://arxiv.org/abs/2110.15538.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation, 2019. URL https://arxiv.org/abs/1812.01754.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training, 2024. URL https://arxiv.org/abs/2305.03053.

Zheng Tang, Chen Liang, Haonan Wu, Lingkai Kong, Juheon Lee, Akshay Krishnamurthy, Eric Xing, and Graham Neubig. Fusionbench: A comprehensive benchmark for deep model fusion. *arXiv:2406.08672*, 2024.

Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers, 2025. URL https://arxiv.org/abs/2506.22712.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Jul 2011.

Mitchell Wortsman, Gabriel Ilharco, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, Ali Farhadi, et al. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022.

Zhengqi Xu, Han Zheng, Jie Song, Li Sun, and Mingli Song. Training-free heterogeneous model merging, 2024. URL https://arxiv.org/abs/2501.00061.

A Related Works

Homogeneous Model Merging Tang et al. (2024) define model merging as the process of fusing the parameters of N similar models into a unified model using a merging function. The process is typically data-efficient, with some methods refining parameters at test time via adaptation or meta-learning. Early work on model merging primarily focused on homogeneous settings in which all participating models share the same architecture, solve the same task, and are trained from the same initialization or pre-training (Wortsman et al., 2022; Matena & Raffel, 2022). This assumption of architectural and task homogeneity made it possible to study merging in relatively controlled environments. Classic findings on linear interpolation and mode connectivity demonstrated that models trained from similar initializations can be joined by low-loss paths, and that simple operations such as weight averaging can already yield competitive performance (Garipov et al., 2018; Draxler et al., 2018; Izmailov et al., 2018; Wortsman et al., 2022).

The focus then expanded to homogeneous architectures trained from different random initializations or partially different data distributions, but that still solve the same task. In this setting, direct weight averaging frequently degrades performance because independently trained networks occupy permutation–equivalent regions of the parameter space. This led to permutation alignment and weight matching approaches, such as re-basin methods, which explicitly search for neuron permutations within the same layer that place models into a shared basin prior to interpolation (Ainsworth et al., 2023). These results showed that many independently trained networks are functionally similar up to symmetries, and that alignment can often recover the benefits of simple averaging.

A further step relaxed the assumption of identical tasks while still keeping architectural similarity. Methods such as ZipIt (Stoica et al., 2024) introduced a "zipping" operation that merges intermediate representations by explicitly pairing and combining features across models at corresponding layers. Since it supports both full and partial zipping, it enables models to be merged only up to a chosen depth. This design allows merging networks in multi-task settings while keeping task-specific heads.

More recent work has explored learning the alignment itself rather than relying solely on fixed similarity heuristics between neurons. PLEAS (Nasery et al., 2025) proposes an optimization-based permutation learning framework that explicitly searches for neuron or channel permutations which minimize functional discrepancy between models before merging. Together, these approaches highlight a shift from purely arithmetic fusion to representation-aware and optimization-driven alignment strategies within homogeneous or near-homogeneous settings.

Heterogeneous Model Merging In heterogeneous merging, architectural mismatch breaks the correspondences assumed by standard align and average pipelines (depth breaks layer correspondences while width breaks neuron correspondences), so recent methods typically (i) construct an explicit cross-model correspondence in representation space, (ii) expand or reduce one model so both admit a compatible layer structure, and only then (iii) apply a homogeneous fusion routine. Nguyen et al. (2023) propose *CLAFusion*, which formulates cross-layer alignment as a constrained one-to-one assignment from layers of the shallower network to layers of the deeper network using representational similarity (specifically linear CKA on activation-based layer representations). Given this mapping, they balance depth either by adding function-preserving layers to the shallower model (in a Net2Net manner, see Appendix C) or by merging/removing layers in the deeper model. They then apply a standard layer-wise fusion method (OTFusion) on the now homogeneous networks, and report that an additional finetuning stage is generally beneficial. In a training-free setting, Xu et al. (2024) perform layer matching by partitioning the deeper model into segments of consecutive layers, choosing segment boundaries to maximize CKA-based similarity between each segment's representation and the corresponding layer in the shallower model (either the average representation of all layers in the segment, or the output representation of the final layer of the segment). They then use the same Net2Net function-preserving expansion procedure and either alignment or ZipIt for merging. For width heterogeneity, they just expand the ZipIt algorithm to accept layers of different widths.

B RESNET VARIANTS USED IN OUR ANALYSIS

Table 1 shows the architectural structure of the different ResNet variants (R17, R29, R35, R50, R101, and R152) used in the experiments. It shows, for each network stage, the output resolution and how many bottleneck residual blocks with specific channel sizes are stacked.

Table 1: Resnets used in our experiments. $\mathrm{B}(c)$ is a bottleneck residual block consisting of conv layers 1×1 with c channels, 3×3 with c channels, and a final 1×1 with $4c$ channels.

stage	out	R17	R29	R35	R50	R101	R152
conv1	112^2	7×7, 64, stride 2					
pool	56^2	3×3 maxpool, stride 2					
conv2_x	56^2	$[\mathrm{B}(64)]\times 1$	$[\mathrm{B}(64)]\times 2$	$[\mathrm{B}(64)]\times 2$	$[\mathrm{B}(64)]\times 3$	$[\mathrm{B}(64)]\times 3$	$[\mathrm{B}(64)]\times 3$
conv3_x	28^2	$[\mathrm{B}(128)]\times 1$	$[\mathrm{B}(128)]\times 2$	$[\mathrm{B}(128)]\times 3$	$[\mathrm{B}(128)]\times 4$	$[\mathrm{B}(128)]\times 4$	$[\mathrm{B}(128)]\times 8$
conv4_x	14^2	$[\mathrm{B}(256)]\times 2$	$[\mathrm{B}(256)]\times 3$	$[\mathrm{B}(256)]\times 4$	$[\mathrm{B}(256)]\times 6$	$[\mathrm{B}(256)]\times 23$	$[\mathrm{B}(256)]\times 36$
conv5_x	7^2	$[\mathrm{B}(512)]\times 1$	$[\mathrm{B}(512)]\times 2$	$[\mathrm{B}(512)]\times 2$	$[\mathrm{B}(512)]\times 3$	$[\mathrm{B}(512)]\times 3$	$[\mathrm{B}(512)]\times 3$
head	1^2	avgpool, FC(1000)					

C NET2NET MATCHING AND EXPANSION PROCEDURE

We use a depth-only Net2Net expansion to embed a shallower ResNet into a deeper topology while preserving the network function at initialization. No training is performed during expansion.

For Bottleneck ResNets with stages conv2_x–conv5_x, let stage $s \in 2, 3, 4, 5$ contain n_s blocks in the shallow model and $N_s \geq n_s$ in the target. Each block computes $y = x + F(x)$.

We instantiate the deeper architecture, copy all name-matching parameters, and append extra blocks at the end of each stage ($i = n_s+1, \ldots, N_s$). These added blocks are identity-initialized ($F(x) \equiv 0$, skip unchanged, stride 1, no projection). The process is strictly positional: that is, we perform no reordering or similarity matching.

The resulting network matches the deeper topology, preserves original parameters and order, and differs only by appended identity blocks, so $\tilde{f}(x) = f_{\text{small}}(x)$ at initialization (up to minor normalization differences) while stage depths and layer indices align.

D PERMUTATION ALIGNMENT DETAILS

We use the `match_tensors_permute` procedure from the ZipIt code repository (Stoica et al., 2024). This is an activation–based permutation alignment method that operates on statistics derived from intermediate forward activations. Given a covariance matrix computed from hooked layer outputs, the routine converts it into a correlation matrix and interprets feature dimensions from each model as nodes to be aligned. The first model is assigned an identity permutation and serves as the reference space. For each additional model, the Hungarian algorithm (through Scipy's `linear_sum_assignment`) is applied to find the channel permutation that maximizes pairwise activation correlation with the reference. The resulting permutation matrices are concatenated into an "unmerge" operator and normalized into a complementary "merge" operator, which are subsequently used to reorder channels before parameter interpolation. Activation statistics are gathered by inserting forward hooks and running a metrics computation pass over a dataset, making the alignment data–dependent but training–free and independent of direct weight matching. The dataset in the ImageNet same-task setting is a subsampled and stratified 100 batches (with batch size 256) of the ImageNet training set, while in the multi-task setting, it is simply both training sets that the models being merged were finetuned on.

E SAME-TASK IMAGENET-IMAGENET MERGING SCORES

Table 2 reports the original measured scores for all ImageNet–ImageNet merging experiments used throughout the analysis. Each row corresponds to merging two independently trained ImageNet

Table 2: Original ImageNet-ImageNet merging results. n is the number of independent runs.

Models	Acc	n
ResNet17 + ResNet17	48.69 ± 0.67	6
ResNet29 + ResNet29	49.71 ± 1.36	6
ResNet35 + ResNet35	51.31 ± 1.18	6
ResNet50 + ResNet50	52.90 ± 1.61	10
ResNet101 + ResNet101	49.26 ± 2.51	3
ResNet152 + ResNet152	46.79 ± 0.68	3
ResNet29 + ResNet17	44.60 ± 0.97	16
ResNet35 + ResNet29	50.24 ± 0.86	16
ResNet35 + ResNet17	45.71 ± 0.91	16
ResNet50 + ResNet35	51.17 ± 1.46	20
ResNet50 + ResNet29	50.05 ± 1.17	20
ResNet50 + ResNet17	49.54 ± 0.62	20
ResNet101 + ResNet50	51.59 ± 1.57	15
ResNet101 + ResNet35	49.41 ± 1.75	12
ResNet101 + ResNet29	49.11 ± 1.35	12
ResNet101 + ResNet17	49.06 ± 1.45	12
ResNet152 + ResNet101	46.88 ± 2.74	9
ResNet152 + ResNet50	48.46 ± 1.59	15
ResNet152 + ResNet35	46.42 ± 2.02	12
ResNet152 + ResNet29	46.08 ± 1.03	12
ResNet152 + ResNet17	45.73 ± 1.34	12

models. Homogeneous rows have depth gap $\Delta d = 0$, while heterogeneous rows have $\Delta d > 0$. We report top-1 accuracy (**Acc**) and the number of runs n (seeds).

F MULTITASK MERGING SCORES

Table 3 reports the original measured scores for all dataset pairs, with homogeneous model pairs in the upper sub-group and heterogeneous pais in the lower sub-group.

Table 3: Model merging results across all dataset pairs. Rows 1–3 per group are *homogeneous* (same architecture); rows 4–6 are *heterogeneous* (mixed architectures). **Acc**: merged accuracy; **Task A/Task B**: individual task accuracies; **Avg**: per-task mean. Values are mean $\pm$ std over multiple runs where available.

Task Pair	**Model A**	**Model B**	**Acc**	**Task A**	**Task B**	**Per-task Avg**
cub, dtd	R50	R50	37.82 ± 2.15	45.15 ± 4.12	39.43 ± 1.32	42.29 ± 1.99
	R101	R101	35.64 ± 1.10	41.04 ± 1.78	38.11 ± 3.19	39.58 ± 1.53
	R152	R152	34.31	40.19	35.63	37.91
	R101	R50	35.48 ± 1.90	38.26 ± 2.64	41.99 ± 1.89	40.13 ± 1.58
	R152	R101	35.07 ± 2.11	39.82 ± 2.99	40.30 ± 0.54	40.06 ± 1.59
	R152	R50	34.45 ± 2.11	37.62 ± 3.11	41.42 ± 1.71	39.52 ± 1.73
dtd, cub	R50	R50	37.41 ± 3.24	42.34 ± 2.34	40.74 ± 3.18	41.54 ± 2.29
	R101	R101	36.63 ± 3.09	39.60 ± 1.14	41.59 ± 2.86	40.60 ± 1.91
	R152	R152	34.55	36.54	38.47	37.50
	R101	R50	34.63 ± 1.35	39.71 ± 1.84	41.77 ± 2.84	40.74 ± 1.63
	R152	R101	34.69 ± 2.09	39.06 ± 2.17	39.31 ± 3.12	39.18 ± 1.96
	R152	R50	32.49 ± 1.09	40.21 ± 2.24	38.54 ± 2.33	39.38 ± 1.28

Continued on next page

Task Pair	Model A	Model B	Acc	Task A	Task B	Per-task Avg
cub, infograph	R50	R50	7.48 ± 1.34	8.20 ± 2.99	10.82 ± 1.48	9.51 ± 0.97
	R101	R101	7.01 ± 0.66	4.38 ± 0.75	9.92 ± 0.84	7.15 ± 0.74
	R152	R152	6.30	5.98	9.09	7.54
	R101	R50	7.04 ± 0.87	4.97 ± 0.63	9.70 ± 1.18	7.33 ± 0.55
	R152	R101	6.20 ± 0.63	7.04 ± 0.99	8.75 ± 0.90	7.89 ± 0.77
	R152	R50	6.41 ± 0.92	6.69 ± 1.13	8.82 ± 1.25	7.76 ± 0.62
infograph, cub	R50	R50	9.61 ± 1.38	13.26 ± 1.83	5.82 ± 1.29	9.54 ± 0.80
	R101	R101	7.45 ± 0.47	10.51 ± 0.57	3.40 ± 1.01	6.95 ± 0.68
	R152	R152	6.98	9.65	6.93	8.29
	R101	R50	7.01 ± 1.04	10.43 ± 0.95	7.12 ± 2.05	8.77 ± 0.79
	R152	R101	7.03 ± 0.47	9.90 ± 0.57	6.21 ± 1.41	8.05 ± 0.89
	R152	R50	6.59 ± 1.14	10.02 ± 0.84	7.24 ± 1.71	8.63 ± 0.73
cub, nabird	R50	R50	36.56 ± 1.05	62.62 ± 2.25	45.40 ± 1.33	54.01 ± 1.44
	R101	R101	36.15 ± 2.04	60.50 ± 1.72	44.53 ± 2.50	52.52 ± 2.09
	R152	R152	32.87	55.22	40.46	47.84
	R101	R50	36.94 ± 2.96	58.90 ± 1.90	45.62 ± 3.66	52.26 ± 2.70
	R152	R101	33.95 ± 1.67	58.25 ± 2.01	41.85 ± 2.08	50.05 ± 2.00
	R152	R50	34.54 ± 2.99	56.96 ± 1.50	42.66 ± 3.69	49.81 ± 2.55
nabird, cub	R50	R50	40.55 ± 3.47	50.03 ± 4.34	61.85 ± 1.92	55.94 ± 3.08
	R101	R101	36.06 ± 1.90	44.48 ± 2.34	60.48 ± 2.69	52.48 ± 2.48
	R152	R152	34.9	43.09	57.17	50.13
	R101	R50	34.82 ± 2.27	43.65 ± 1.71	60.83 ± 1.85	52.24 ± 1.49
	R152	R101	35.21 ± 2.19	43.40 ± 2.70	58.80 ± 2.31	51.10 ± 2.45
	R152	R50	32.83 ± 2.15	41.30 ± 1.53	58.10 ± 1.95	49.70 ± 1.41
cub, svhn	R50	R50	47.65 ± 9.70	1.49 ± 0.50	68.12 ± 5.39	34.80 ± 2.60
	R101	R101	48.62 ± 5.38	1.18 ± 0.19	69.03 ± 1.53	35.11 ± 0.86
	R152	R152	41.86	1.74	62.42	32.08
	R101	R50	47.42 ± 4.87	1.18 ± 0.22	61.24 ± 4.53	31.21 ± 2.26
	R152	R101	47.81 ± 1.83	1.44 ± 0.50	66.36 ± 1.34	33.90 ± 0.52
	R152	R50	44.02 ± 4.51	1.41 ± 0.36	56.96 ± 4.01	29.18 ± 1.89
svhn, cub	R50	R50	59.49 ± 6.09	75.08 ± 5.86	1.07 ± 0.20	38.07 ± 2.91
	R101	R101	50.99 ± 3.41	68.25 ± 4.99	1.80 ± 0.17	35.02 ± 2.56
	R152	R152	50.02	68.57	1.63	35.10
	R101	R50	45.67 ± 11.22	73.49 ± 5.41	1.69 ± 0.37	37.59 ± 2.69
	R152	R101	46.30 ± 3.56	66.55 ± 2.84	1.75 ± 0.23	34.15 ± 1.44
	R152	R50	46.78 ± 10.24	75.25 ± 4.06	1.82 ± 0.46	38.53 ± 1.96
eurosat, mnist	R50	R50	43.84 ± 8.51	44.54 ± 9.09	83.69 ± 5.92	64.12 ± 6.64
	R101	R101	37.53 ± 1.79	39.72 ± 11.05	87.18 ± 3.39	63.44 ± 6.63
	R152	R152	48.18	39.03	88.88	63.95
	R101	R50	46.37 ± 10.78	45.95 ± 8.22	83.71 ± 8.58	64.83 ± 4.52
	R152	R101	38.47 ± 8.24	47.11 ± 4.89	85.74 ± 3.97	66.42 ± 2.06
	R152	R50	50.01 ± 12.09	49.75 ± 7.82	86.05 ± 6.34	67.90 ± 5.14
mnist, eurosat	R50	R50	55.21 ± 12.46	90.28 ± 4.71	44.13 ± 8.40	67.20 ± 4.14
	R101	R101	42.97 ± 4.39	85.77 ± 4.77	48.60 ± 6.73	67.18 ± 3.43
	R152	R152	42.25	92.08	50.12	71.10
	R101	R50	46.79 ± 5.33	93.48 ± 1.95	48.82 ± 9.35	71.15 ± 4.69
	R152	R101	44.38 ± 4.21	91.09 ± 2.84	45.70 ± 8.20	68.40 ± 4.46
	R152	R50	46.49 ± 7.77	93.79 ± 2.29	48.96 ± 8.49	71.37 ± 4.92

Continued on next page

Task Pair	Model A	Model B	Acc	Task A	Task B	Per-task Avg
mnist, svhn	R50	R50	38.44 ± 6.34	96.03 ± 1.22	56.33 ± 4.58	76.18 ± 1.91
	R101	R101	44.05 ± 0.18	94.73 ± 0.29	61.19 ± 2.58	77.96 ± 1.32
	R152	R152	42.90	93.43	66.92	80.17
	R101	R50	46.51 ± 1.59	96.67 ± 0.70	52.46 ± 3.48	74.56 ± 1.65
	R152	R101	48.62 ± 1.59	96.06 ± 0.43	66.08 ± 3.31	81.07 ± 1.68
	R152	R50	48.17 ± 2.33	96.00 ± 0.94	55.98 ± 3.61	75.99 ± 1.83
svhn, mnist	R50	R50	49.52 ± 3.86	65.81 ± 4.91	94.41 ± 0.86	80.11 ± 2.36
	R101	R101	45.42 ± 2.31	62.63 ± 2.66	95.53 ± 0.82	79.08 ± 1.35
	R152	R152	45.27	69.87	95.32	82.60
	R101	R50	41.38 ± 4.59	75.92 ± 6.04	95.18 ± 1.09	85.55 ± 2.80
	R152	R101	44.92 ± 1.72	66.73 ± 0.74	94.87 ± 0.79	80.80 ± 0.57
	R152	R50	42.45 ± 4.62	81.68 ± 4.02	95.17 ± 1.32	88.42 ± 2.09

G Dataset similarity using FID-style Metric

Given two image datasets (S) and (T), we define a distance between their distributions in a semantic feature space extracted by a pretrained vision network. We specifically use the DINOv2 feature encoder (Oquab et al., 2024). We compute embeddings for all samples: $\{f(x)\}_{x \in S}$ and $\{f(y)\}_{y \in T}$, and fit multivariate Gaussians to each embedded sample set using the empirical mean and covariance:

$$\mu_S = \frac{1}{|S|} \sum_{x \in S} f(x), \quad \Sigma_S = \frac{1}{|S| - 1} \sum_{x \in S} (f(x) - \mu_S)(f(x) - \mu_S)^\top$$

We similarly compute (μ_T, Σ_T). Heusel et al. (2018) define FID as the Fréchet distance between the two fitted Gaussians, which has the closed form:

$$\mathrm{FID}(S, T) = \|\mu_S - \mu_T\|_2^2 + \mathrm{tr}\left(\Sigma_S + \Sigma_T - 2(\Sigma_S \Sigma_T)^{1/2}\right).$$

This exact formula for multivariate normal distributions is given by Dowson & Landau (1982). We convert the distance into a bounded similarity score using an exponential kernel,

$$\mathrm{Sim}(S, T) = \exp\left(-\frac{\mathrm{FID}(S, T)}{\tau}\right),$$

where $\tau > 0$ is a temperature parameter controlling how quickly similarity decays with distance.

The resulting pairwise similarities between datasets are summarized in Table 4, showing that CUB and NABirds exhibit the highest semantic similarity (which is logical, as they are both bird datasets) while MNIST and EuroSAT are the least similar among the evaluated pairs.

Table 4: Dataset Similarities (FID-style Metric)

Dataset A	Dataset B	Similarity
CUB	NABIRD	0.78586
MNIST	SVHN	0.42535
DTD	CUB	0.33306
CUB	DomainNet Infograph	0.32415
SVHN	CUB	0.25578
MNIST	EuroSAT	0.23152