

---

# ZipIt!: Multitask Model Merging *without Training*

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

1 We tackle the extremely difficult problem of combining distinct models with  
2 different initializations, each solving a separate task, into one multi-task model  
3 **without any additional training**. Prior work in model merging permutes one  
4 model to the space of the other then averages them together. While this works for  
5 models trained on the same task, we find that this fails to account for the differences  
6 in models trained on disjoint tasks. Thus, we introduce “ZipIt!”, a general method  
7 for merging two arbitrary models of the same architecture that incorporates two  
8 simple strategies. First, in order to account for features that aren’t shared between  
9 models, we expand the model merging problem to allow for merging features  
10 *within* each model by defining a general “zip” operation. Second, we add support  
11 for *partially zipping* the models up until a specified layer, naturally creating a multi-  
12 head model. We find that these two changes combined account for a staggering  
13 20-50% improvement over prior work,

## 14 1 Introduction

15 Combining multiple models into one *without training* has recently started to gain traction in the  
16 vision community. Model Soups [1] can add multiple models finetuned from the same pretrained  
17 initialization to improve accuracy and robustness. Git Re-Basin [2] generalizes further to models  
18 trained on the same data but with different initializations, though with a significant accuracy drop.  
19 REPAIR [3] improves on Git Re-Basin by adding new parameters and adjusting model batch norms  
20 where applicable. However, all of these methods only combine models trained on the same task.  
21 In this paper, we take this line of work to a logical extreme: merging differently initialized models  
22 trained on *completely separate* tasks (see Fig. 1ab). We show that this is an incredibly difficult  
23 problem for prior work and employ two simple strategies to make it feasible.

24 First, prior work focuses on *permuting* one model to the other when merging them. This inherently  
25 assumes that most features *across* them are correlated. But, this is not always the case for models  
26 trained on different tasks. Instead, we generalize model merging to support “zipping” any combination  
27 of correlated features *within* and *across* each model. We find that on some tasks, this alone improves  
28 accuracy **by up to 20%** vs. permutation-based approaches

29 Second, the features of models trained on disjoint tasks become less correlated over the course of the  
30 network [4]. Thus, we introduce *partial zipping*, which allows us to only “zip” up to a specified layer.  
31 Afterward, we feed the merged model’s outputs to the remaining unmerged layers of the original  
32 networks, creating a multi-head model. Partially zipping can improve accuracy **by over 15%**.

33 ZipIt! (Fig. 1c) incorporates both strategies to “zip” models trained on different tasks into a single  
34 multitask model *without retraining*. ZipIt! is general and supports merging arbitrary models of the  
35 same architecture together (Sec. 3). We validate ZipIt! by merging models trained on different tasks  
36 (including classification and segmentation) into one, significantly outperforming prior work (Sec. 4).

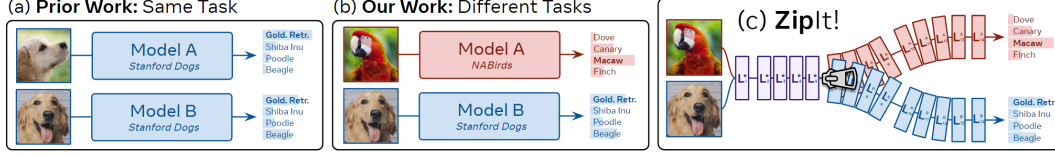


Figure 1: **Setting and ZipIt!** (a) Prior work merges models from the **same** dataset with the **same** label sets: e.g., merging two models both trained to classify dog breeds. (b) Our setting expands to merging models from **different** datasets with **different** label sets: e.g., merging a model that classifies dog breeds with one that classifies bird species. (c) **ZipIt!** merges these models *without retraining* by identifying shared features. Depending on the task, ZipIt! can nearly match ensemble performance.

## 2 Related Work

Model merging combines the weights of two or more models into a one. Our work differs from prior work in that we merge differently initialized models trained on disjoint tasks (Fig. 1) without training.

**Merging Finetuned Models.** If two models are finetuned from the same pretrained checkpoint, they often lie in the same error minima [5]. [6, 7, 8, 9, 10, 11] have exploited this property to average together the weights of a model at different stages of training. [12, 13, 14, 15, 16] use an “exponential moving average” of training checkpoints as a teacher for additional self-supervised learning. Other works, [1, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27] merge models fully finetuned to improve performance on a task. Our setting differs, as we do not assume the same initialization.

**Merging Differently Initialized Models.** Works in this space often merge models trained on the same task and rely on mode connectivity [28, 29, 30, 31, 32, 33, 34], as differently initialized models may not lie in the same error minima [29, 35]. Most recent work follows the intuition formalized by [35] that models permuted to the same loss minima can be merged by averaging their weights [35, 2, 3, 36]. Similar to ZipIt! [37] merges models of different tasks, but requires jointly finetuning on all tasks after each layer merge. As far as we are aware, we present the first *general method* to successfully merge models trained on disjoint tasks *without additional training*.

## 3 ZipIt!

In this work, we treat model merging as combining the checkpoints (i.e., collection of weights) of multiple models layer-by-layer into a single checkpoint that can perform all the tasks of its constituents. Consider a model  $\mathcal{L}$  as a collection of layers  $L_i \in \mathcal{L}$ , each of which may have some parameters (e.g.,  $W_i, b_i$  for a linear layer). Suppose  $L_i \in \mathcal{L}$  is a linear layer with parameters  $W_i \in \mathbb{R}^{n_i \times m_i}, b_i \in \mathbb{R}^{n_i}$  with input features  $x \in \mathbb{R}^{m_i}$  and outputs features  $f_i \in \mathbb{R}^{n_i}$ . Then,  $f_i = L_i(x) = W_i x + b_i$ .

Our goal is to take  $L_i^A \in \mathcal{L}^A$  from a model A and  $L_i^B \in \mathcal{L}^B$  from a model B and merge them into a layer  $L_i^*$  that combines their feature spaces such that information from both  $f_i^A$  and  $f_i^B$  is retained in  $f_i^*$ . We accomplish this by merging each layer of one model with the corresponding layer in the other, both merging features *across* both layers and *within* the same layer. This is in contrast to permutation-based merging method, which only combine features *across* layers.

**Problems with Permutation.** Permutation methods posit that model B can be moved to the same loss minima as model A via permutation with high-likelihood [35]. However, this is unlikely when models are trained on different tasks because each model optimizes for task-specific minimas. In this case the optimal permutation of model B to model A lies in a strong minima on task B but may not lie in a minima on task A, as shown in Fig. 2. This causes the interpolated model to perform worse than either of the two original models. Thus, we explore alternative merging methods.

**Why should we merge *within*?** Features of models trained on different tasks may be dissimilar, as the models solve different problems. Instead, those features may be more compatible with others within the same model, which would better retain performance when combined.

**What is our merge strategy?** Prior work obtains  $f_i^*$  by combining one feature from  $f_i^A$  and one from  $f_i^B$ . [38, 3] determine which features to merge by computing the pairwise correlations between

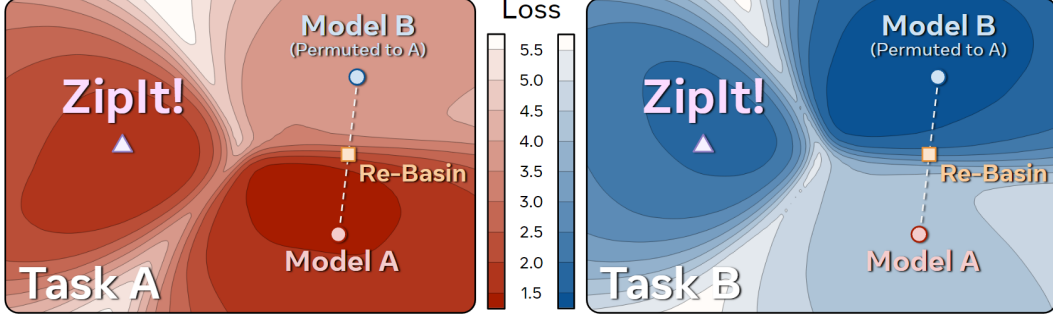


Figure 2: **Task Loss Landscapes** for models in Tab. 1b. **Model A** and **Model B** lie in low loss minimas for their own tasks, but *not for the other task*. Thus, any interpolation between **Model A** and a permuted **Model B** (e.g., Git Re-basin) lies outside the minima *for both tasks* and thus performs poorly. In contrast, ZipIt! improves the merge by finding a model that lies in a loss minima for both.

the neuron activations of  $f_i^A$  and  $f_i^B$  over a set of images. In contrast, our approach can also obtain  $f_i^*$  by combining two features from just  $f_i^A$  or  $f_i^B$ . We compute the same correlations, but also include those of  $f_i^A$  and  $f_i^B$  each with themselves over the same data. We then greedily choose without replacement the pairs with highest correlation and average their features to obtain  $f_i^*$ .

**Merging a layer.** For every layer, we first concatenate  $f_i^A$  and  $f_i^B$  into  $f_i^A || f_i^B \in \mathbb{R}^{2n_i}$ , and define a “merge matrix”  $M_i \in \mathbb{R}^{n_i \times 2n_i}$  based on the matches from our greedy algorithm (i.e.,  $f_i^* = M_i (f_i^A || f_i^B)$ ). Thus,  $M_i$  merges the output spaces of  $L_i^A$ , and  $L_i^B$  into  $L_i^*$ . We then define an “unmerge” matrix  $U_i \in \mathbb{R}^{2n_i \times n_i}$  s.t. it *undoes* the merge operation:  $U_i f_i^* \approx f_i^A || f_i^B$ . Applying  $U_i$  ensures the input spaces of the future layers  $L_{i+1}^A$ , and  $L_{i+1}^B$  are aligned with current merged layer  $L_i^*$ . Once all matrices are computed at each layer, we obtain  $L_i^*$  for a linear layer by

$$W_i^* = M_i \begin{pmatrix} W_i^A & 0 \\ 0 & W_i^B \end{pmatrix} U_{i-1} \quad b_i^* = M_i \begin{pmatrix} b_i^A \\ b_i^B \end{pmatrix}.$$

Due to space restrictions, please see Appendix. A for more details on merging other kinds of layers.

**Partial Zipping.** Sometimes, later layers in the networks have very uncorrelated (dissimilar) outputs. Forcibly zipping these would lead to meaningless features. In this case, we can perform a *partial zip*. That is, we zip most of the layers together, but leave the later ones *unzipped*—obtaining a multihead model.

**Merging Many Models ( $\alpha$ ).** Sometimes, we’d like to merge more than two models together. To do this, we allow “repeated matches”: we replace matched features from our algorithm with the resulting merged feature instead of removing them completely. To ensure that one feature doesn’t get merged endlessly, we set the correlations of the new feature to be the minimum of the old features’ similarities weighted by  $\alpha \in (0, 1]$ . We find a small value of  $\alpha$  typically works best.

**Within-Model Merging Budget ( $\beta$ ).** To demonstrate the effectiveness of same-model merges, we introduce a “budget” parameter  $\beta \in [0, 1]$  that denotes what percent of total merged features can come from models merging within themselves, with each model receiving an equal portion of this budget. A budget of 0 only allows feature merging across models and yields a permutation-merge.

## 4 Results

There’s no standard benchmark to evaluate merging approaches on models from distinct tasks, so we construct our own. We evaluate our approach in two different settings. (1) A versatile test-bed: disjoint category splits of the same dataset (i.e., *same dataset and different label sets*). (2) A very challenging setting: completely different datasets and tasks (i.e., *different datasets and label sets*).

We compare to three baselines: Git Re-Basin [2], Weight Averaging (W. Avg) and Permute. W. Avg involves simply average all model parameters to be merged, while we design Permute to use linear sum assignment to find optimal permutations (following [38]) for merging. Note that Permute is a

Method	FLOPs (G)	Accuracies (%)			
		Joint	Task A	Task B	Avg
Model A	0.68	48.2 $\pm$ 1.0	97.0 $\pm$ 0.6	45.1 $\pm$ 8.6	71.0 $\pm$ 4.4
Model B	0.68	48.4 $\pm$ 3.8	49.1 $\pm$ 9.3	96.1 $\pm$ 1.1	72.6 $\pm$ 4.9
W. Avg	0.68	43.0 $\pm$ 1.6	54.1 $\pm$ 1.4	67.5 $\pm$ 1.2	60.8 $\pm$ 4.5
Git Re-Basin <sup>†</sup>	0.68	46.2 $\pm$ 0.8	76.8 $\pm$ 8.9	82.7 $\pm$ 5.1	79.8 $\pm$ 6.5
Permute	0.68	58.4 $\pm$ 6.8	86.6 $\pm$ 2.1	87.4 $\pm$ 1.1	87.4 $\pm$ 1.4
<b>ZipIt!<sub>20/20</sub></b>	0.68	<b>79.1<math>\pm</math>1.1</b>	<b>92.9<math>\pm</math>1.1</b>	<b>91.2<math>\pm</math>1.4</b>	<b>92.1<math>\pm</math>1.0</b>
Ensemble	1.37	87.4 $\pm$ 2.6	97.0 $\pm$ 0.6	96.1 $\pm$ 1.1	96.6 $\pm$ 0.4
<b>ZipIt!<sub>13/20</sub></b>	0.91	<b>83.8<math>\pm</math>3.1</b>	<b>95.1<math>\pm</math>0.7</b>	<b>94.1<math>\pm</math>1.5</b>	<b>94.6<math>\pm</math>0.6</b>

(a) **CIFAR-10 (5+5)**. ResNet-20 (4 $\times$  width).

Method	FLOPs (G)	Accuracies (%)			
		Joint	Task A	Task B	Avg
Model A	2.72	41.6 $\pm$ 0.3	82.9 $\pm$ 0.7	24.8 $\pm$ 0.4	53.9 $\pm$ 0.5
Model B	2.72	41.6 $\pm$ 0.2	25.1 $\pm$ 1.2	82.8 $\pm$ 0.2	54.0 $\pm$ 0.6
W. Avg	2.72	17.0 $\pm$ 1.7	23.8 $\pm$ 6.9	24.8 $\pm$ 5.9	24.3 $\pm$ 1.9
Git Re-Basin <sup>†</sup>	2.72	40.9 $\pm$ 0.2	57.3 $\pm$ 1.5	56.7 $\pm$ 0.7	57.0 $\pm$ 0.8
Permute	2.72	42.8 $\pm$ 0.7	61.6 $\pm$ 1.4	60.5 $\pm$ 0.5	61.0 $\pm$ 0.8
<b>ZipIt!<sub>20/20</sub></b>	2.72	<b>54.9<math>\pm</math>0.8</b>	<b>68.2<math>\pm</math>0.8</b>	<b>67.9<math>\pm</math>0.6</b>	<b>68.0<math>\pm</math>0.4</b>
Ensemble	5.45	73.5 $\pm$ 0.4	82.9 $\pm$ 0.7	82.8 $\pm$ 0.3	82.8 $\pm$ 0.4
<b>ZipIt!<sub>13/20</sub></b>	3.63	<b>70.2<math>\pm</math>0.4</b>	<b>80.3<math>\pm</math>0.8</b>	<b>80.1<math>\pm</math>0.7</b>	<b>80.2<math>\pm</math>0.6</b>

(b) **CIFAR-100 (50+50)**. ResNet-20 (8 $\times$  width).

Table 1: **CIFAR Results**. ZipIt! vs. baselines on combining a model trained on half the classes (**Task A**) with one trained on the other half (**Task B**) *without extra training*. We report both joint (10/100-way) and per-task (5/50-way) accuracy. ZipIt! *significantly* outperforms its baseline and closes in on the upper bound (ensemble accuracy). <sup>†</sup> refers to [2].

107 *strong* baseline we create and is more accurate than Git Re-Basin in our settings. For our method,  
108 ZipIt!<sub>n/m</sub> indicates that  $n$  out of the  $m$  layers in the network have been zipped (Sec. 3). Note, all our  
109 models have *different initializations*.

110 **Disjoint Category Splits**. In Tab. 1a, we merge five pairs of models trained on disjoint 5 class subsets  
111 of CIFAR-10 using ResNet-20 with a 4 $\times$  width multiplier (denoted as ResNet-20 $\times$ 4). We train with  
112 a CLIP-style loss [39] using CLIP text encodings of the class names as targets so that both models  
113 output into the same CLIP-space regardless of the category (required for [2]). We report: (1) joint  
114 accuracy—the accuracy of each model over *all* classes across datasets, and (2) per task accuracy—the  
115 accuracy of each task individually and also their average. Overall, ZipIt! performs a staggering 20.7%  
116 better than the nearest baseline. If we allow the last stage of the network to remain unzipped (i.e., zip  
117 up to 13 layers), our method obtains 83.8%, which is only 3.6% behind an ensemble of **model A** and  
118 **model B** (which is practically the upper bound for this setting). We find similar results on disjoint  
119 50 class splits of CIFAR-100 in Tab. 1b using an 8 $\times$  width multiplier. ZipIt! again significantly  
120 outperforms baselines.

121 **Different Datasets**. We merge ResNet-50 mod-  
122 els trained on: Stanford Dogs [40], Oxford Pets  
123 [41], CUB200 [42], and NABirds [43]. In Tab. 2,  
124 we show the average per task accuracy from ex-  
125 haustively merging each pair and the much more  
126 difficult setting of merging all four at once. We  
127 report the accuracy of our baselines by applying  
128 them up until the last layer, but we can’t compare  
129 to [2] as it requires shared output space.

130 For pairs of models, ZipIt! slightly outperforms  
131 our permute baseline across all tasks and performs  
132 similarly when merging all 4 models at once. How-  
133 ever, if we add capacity to the merged model  
134 through partial zipping, we perform up to 33%  
135 better on merging pairs and 50% better on merg-  
136 ing all four models than the permute baseline.

137 We also merge the ResNet-50 backbone of a  
138 DeeplabV3 [44] segmentation model that achieves  
139 76.8% mIoU on PASCAL VOC [45] with an ImageNet-1k [46] classification model that obtains  
140 77.8% accuracy. ZipIt! can still achieve good performance on *both* tasks even with half of layers  
141 merged: obtaining 64.4% mIoU on PASCAL VOC and 60.9% accuracy on ImageNet-1k.

## 142 5 Conclusion

143 In this paper, we tackle the extremely difficult task of merging models trained on completely disjoint  
144 tasks *without additional training*. We find that prior work underperforms in this setting and posit  
145 that they neither fully (1) exploit model similarities nor (2) account for model dissimilarities. We  
146 introduce ZipIt!, a general framework for merging models that addresses these issues, and show it to  
147 significantly outperform prior work across several difficult settings.

Method	FLOPs (G)	Per-Task Accuracies (%)				
		SD	OP	CUB	NAB	Avg
Merging Pairs						
W. Avg	4.11	12.9	18.2	13.9	0.2	11.3
Permute	4.11	46.2	47.6	35.6	13.5	35.7
<b>ZipIt!</b> <sub>49/50</sub>	4.11	<b>46.9</b>	<b>50.7</b>	<b>38.0</b>	12.7	<b>37.1</b>
Ensemble	8.22	72.7	81.1	71.0	77.2	75.5
<b>ZipIt!</b> <sub>22/50</sub>	6.39	62.6	71.2	62.8	53.0	62.4
<b>ZipIt!</b> <sub>10/50</sub>	7.42	<b>66.5</b>	<b>75.8</b>	<b>65.6</b>	<b>66.8</b>	<b>68.7</b>
Merging All 4						
W. Avg	4.12	0.8	3.0	0.6	0.3	1.2
Permute	4.12	15.7	26.1	<b>14.0</b>	<b>5.3</b>	15.3
<b>ZipIt!</b> <sub>49/50</sub>	4.12	<b>21.1</b>	<b>33.3</b>	8.6	3.9	<b>16.8</b>
Ensemble	16.4	72.7	81.2	71.0	77.2	75.5
<b>ZipIt!</b> <sub>22/50</sub>	11.0	50.2	55.9	44.0	32.0	45.5
<b>ZipIt!</b> <sub>10/50</sub>	14.1	<b>63.5</b>	<b>70.8</b>	<b>63.7</b>	<b>63.1</b>	<b>65.3</b>

Table 2: **Multi-Dataset Results**. Merging ResNet-50 models trained on *completely different datasets*: Stanford Dogs (SD), Oxford Pets (OP), CUB200 (CUB), and NABirds (NAB). We report average per-task accuracy over merging model pairs, and all four.

## References

- [1] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022.
- [2] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv:2209.04836*, 2022.
- [3] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair. *arXiv:2211.08403*, 2022.
- [4] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.
- [5] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *NeurIPS*, 2020.
- [6] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv:1704.00109*, 2017.
- [7] Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *UAI*, 2018.
- [8] Johannes Von Oswald, Seijin Kobayashi, Joao Sacramento, Alexander Meulemans, Christian Henning, and Benjamin F Grewe. Neural networks with late-phase weights. *arXiv:2007.12927*, 2020.
- [9] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022.
- [10] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. *NeurIPS*, 2022.
- [11] Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, and Leshem Choshen. COLD fusion: Collaborative descent for distributed multitask finetuning. Association for Computational Linguistics, 2023.
- [12] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *NeurIPS*, 2017.
- [13] Zhaowei Cai, Avinash Ravichandran, Subhansu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *CVPR*, 2021.
- [14] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 2020.
- [15] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
- [16] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *ICML*, 2022.
- [17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 2017.
- [18] Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining. *arXiv:2204.03044*, 2022.

- [19] Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Recycling diverse models for out-of-distribution generalization. *arXiv:2212.10445*, 2022.
- [20] Stephen Ashmore and Michael Gashler. A method for finding similarity between multi-layer perceptrons by forward bipartite alignment. In *IJCNN*, 2015.
- [21] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *ICML*, 2019.
- [22] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv:2002.06440*, 2020.
- [23] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv:2212.04089*, 2022.
- [24] Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. Knowledge is a region in weight space for fine-tuned language models. *arXiv:2302.04863*, 2023.
- [25] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving interference when merging models. *arXiv:2306.01708*, 2023.
- [26] Yi-Lin Sung, Linjie Li, Kevin Lin, Zhe Gan, Mohit Bansal, and Lijuan Wang. An empirical study of multimodal model merging, 2023.
- [27] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. *arXiv:2111.09832*, 2021.
- [28] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. *arXiv:1611.01540*, 2016.
- [29] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *NeurIPS*, 2018.
- [30] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *ICML*, 2018.
- [31] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *ICML*, 2020.
- [32] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *NeurIPS*, 2020.
- [33] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. *NeurIPS*, 2020.
- [34] Chang Liu, Chenfei Lou, Runzhong Wang, Alan Yuhan Xi, Li Shen, and Junchi Yan. Deep neural network fusion via graph matching with applications to model ensemble and federated learning. In *ICML*, 2022.
- [35] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv:2110.06296*, 2021.
- [36] Fidel A Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit sinkhorn differentiation. *arXiv:2212.12042*, 2022.
- [37] Xiaoxi He, Zimu Zhou, and Lothar Thiele. Multi-task zipping via layer-wise neuron sharing. *NeurIPS*, 2018.
- [38] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent Learning: Do different neural networks learn the same representations? *arXiv:1511.07543*, 2015.

- [39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- [40] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.
- [41] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.
- [42] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.
- [43] 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 *CVPR*, 2015.
- [44] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.
- [46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [47] Thomas Uriot and Dario Izzo. Safe crossover of neural networks through neuron alignment. In *GECCO*, 2020.
- [48] Aditya Kumar Akash, Sixu Li, and Nicolás García Trillos. Wasserstein barycenter-based model fusion and linear mode connectivity of neural networks. *arXiv:2210.06671*, 2022.
- [49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 2017.
- [50] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.
- [51] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*, 2017.
- [52] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.
- [53] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.
- [54] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV*, 2019.
- [55] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022.
- [56] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2018.
- [57] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.
- [58] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017.
- [59] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, 2021.



- [60] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *TKDE*, 2022.
- [61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2016.
- [62] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [63] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv:1803.03635*, 2018.
- [64] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv:2010.11929*, 2020.
- [65] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019.
- [66] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, 1967.
- [67] Louis L McQuitty. Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies. *Educational and psychological measurement*, 1957.
- [68] Wei-Hong Li and Hakan Bilen. Knowledge distillation for multi-task learning. In *ECCV*, 2020.
- [69] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.
- [70] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022.
- [71] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. *arXiv:2302.05442*, 2023.
- [72] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.
- [73] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *ICML*, 2013.
- [74] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *NeurIPS*, 2011.
- [75] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [76] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [77] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.
- [78] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, 2008.
- [79] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *ICLR*, 2023.



- 330 [80] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon.  
331 Ss-il: Separated softmax for incremental learning. In *ICCV*, 2021.
- 332 [81] Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clement Hongler, Wulfram  
333 Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural  
334 networks: Symmetries and invariances. In *Proceedings of the 38th International Conference on*  
335 *Machine Learning*. PMLR, 2021.
- 336 [82] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and  
337 generalization in neural networks (invited paper). In *Proceedings of the 53rd Annual ACM*  
338 *SIGACT Symposium on Theory of Computing*, 2021.

## 339 A Layer Merging Details

340 Given  $M_i$  and  $U_i$ , here we describe rules more concretely for each layer type needed to define most  
341 convnets.

342 **Conv.** Apply  $M_i$  and  $U_i$  to each kernel location (i.e., move the  $k \times k$  kernel dimensions to the batch  
343 dimension).

344 **BatchNorm.** Apply  $M_i$  to all parameters (weight, bias, mean, variance), squaring it for the variance  
345 term. Continue propagation. As [3] points out, we cannot compute the correct variance without  
346 knowing the covariance between the two models (which we don't have access to). Thus, we reset  
347 batch norms after merging to evaluate the variance correctly.

348 **LayerNorm.** Apply  $M_i$  to all parameters (weight, bias). Since LayerNorm computes mean and  
349 standard deviation on the fly, we don't need to do anything special.

350 **Avg / Max Pool.** Skip.

351 **Skip Connection.** Apply the same  $M_i$  to each layer whose outputs are combined via the residual so  
352 that their output spaces are all aligned.