Semi-Ensemble: A Simple Approach to Over-parameterized Model Interpolation

Jiwoon Lee & Jaeho Lee
Department of Electrical Engineering
Pohang University of Science and Technology
Pohang, South Korea
{jwlee9702, jaeho.lee}@postech.ac.kr

Editors: Marco Fumero, Emanuele Rodolà, Clementine Domine, Francesco Locatello, Gintare Karolina Dziugaite, Mathilde Caron

Abstract

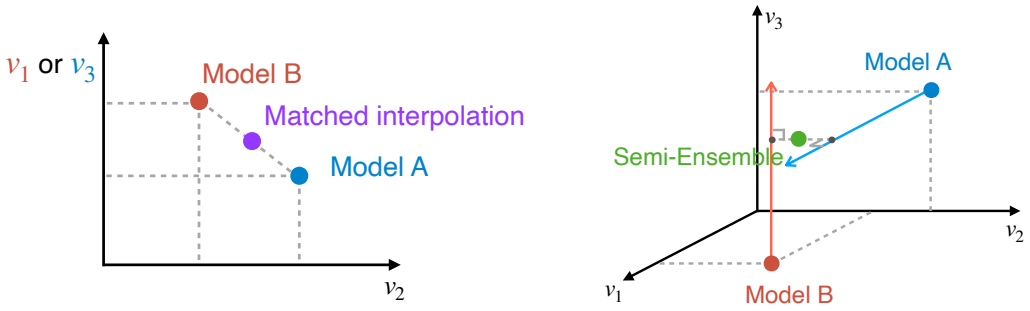
We develop a unified framework for interpolating two models with various degrees of over-parameterization, having model merging and model ensemble as special cases. Instead of directly interpolating models in their original parameter space, the proposed Semi-Ensemble interpolates the over-parameterized versions of the models in a higher-dimensional joint parameter space. Here, the over-parameterizations recover each endpoint model when projected to some low-dimensional subspace spanned by a fraction of bases. By carefully constructing the joint parameter space, the interpolated model can achieve a smooth tradeoff between the total number of parameters and the model accuracy, outperforming existing baselines. Intriguingly, we show that Semi-ensembles can sometimes achieve a better performance than vanilla ensembles, even with a slightly smaller number of parameters.

1 Introduction

Recent breakthroughs in *linear mode connectivity* have given rise to a new paradigm in utilizing the knowledge of two (or more) independently trained models [26, 8]. Unlike typical approaches, e.g., ensembling the model predictions [6], one can instead linearly interpolate the endpoint models in their parameter space to construct a new model. By carefully permuting the order of neurons before mapping to the parameter space, the interpolated model can convey the knowledge of both models *without additional training*. For example, interpolating the endpoint models trained on biased data splits can generate a model that has an improved test accuracy on balanced datasets [29, 26, 1, 17]. Such capability makes this technique, called *model merging*, a promising tool for various practical applications, such as federated learning under communication and privacy constraints [24].

However, it is still in question whether the parameter interpolation has the capability to preserve the "full" knowledge of endpoint models. Even when the interpolated model has a similar or better accuracy than the endpoint models, it tends to perform considerably worse than the ensemble of endpoint models [1, 17]. This performance gap between the merged model and ensemble suggests that some task-relevant information might have been lost during the interpolation. It is often argued that such loss may be an inevitable consequence of having an under-parameterized model as an interpolation [1]; comparing with the collection of endpoint models, the model ensemble has the same number of parameters, whereas the merged model has only half. This perspective suggests that the information loss may be able to be mitigated if one can generate an *over-parameterized interpolation* of models. Unfortunately, however, it is not yet clear how this can be done.

Proceedings of the I edition of the Workshop on Unifying Representations in Neural Models (UniReps 2023).



(a) Model interpolation without over-parameterization

(b) Semi-Ensemble

Figure 1: **Concept of Semi-Ensemble**. The semi-ensemble interpolates the endpoint models while maintaining the information of inherent parameters. When Model A is trained to have weights $(v_2^{\mathsf{A}}, v_3^{\mathsf{A}})$ and Model B is trained to have weights $(v_1^{\mathsf{B}}, v_2^{\mathsf{B}})$, existing model interpolation methods without any over-parameterization give low-dimensional weights $(\frac{v_2^{\mathsf{A}}+v_2^{\mathsf{B}}}{2}, \frac{v_3^{\mathsf{A}}+v_1^{\mathsf{B}}}{2})$, where the second element may contain no useful information. The proposed Semi-Ensemble preserves unique features by interpolating the models lifted to the higher-dimensional joint parameter space, giving $(v_1^{\mathsf{B}}, \frac{v_2^{\mathsf{A}}+v_2^{\mathsf{B}}}{2}, v_3^{\mathsf{A}})$.

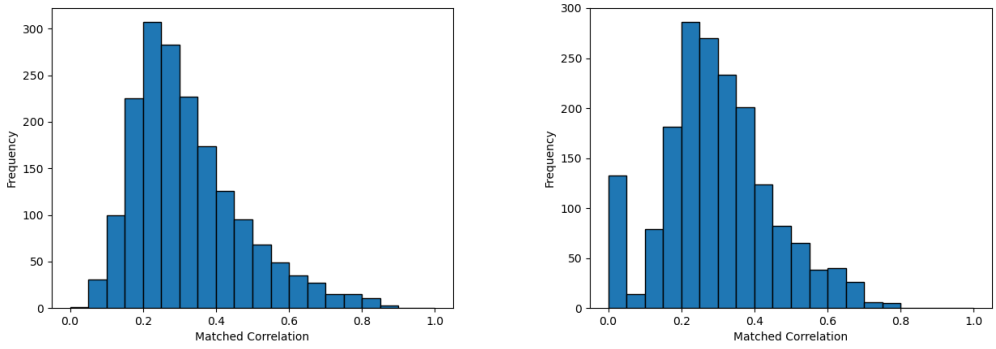
In this paper, we develop a new framework to describe model merging and model ensemble under a unified perspective (see Figure 1b). Simply put, we consider an interpolation of over-parameterized models that lie in a high-dimensional *joint parameter space* (spanned by $\{v_1, v_2, v_3\}$). The joint parameter space is constructed using the endpoint model parameters as bases, where the parameters from each model can be mapped to the shared bases (v_2) or unique bases (v_1, v_3). The over-parameterized models to be interpolated are elements of two distinct hyperplanes in this space (colored solid lines). Each hyperplane is constructed by treating a corresponding endpoint model as a low-dimensional projection of an over-parameterized model, i.e., a subnetwork generated by pruning a larger neural network [10]. From the hyperplanes, we select the minimum-distance pair of models (small black dots) and then interpolate them to get a new over-parameterized model.

This framework can cover a wide range of scenarios with diverse degrees of over-parameterization, including model merging and ensemble as special cases. Model merging corresponds to the case of no over-parameterization, where the joint parameter space is identical to the parameter space of endpoint models. Model ensemble corresponds to the case of full over-parameterization, where the joint parameter space is a Cartesian product of the endpoint parameter spaces. Projecting the ensemble to each low-dimensional subspace recovers the endpoint models, meaning that the hyperplanes for each endpoint model have a zero distance; thus, no interpolation is required in this case.

By varying degrees of over-parameterization, our framework provides a family of over-parameterized interpolations. We show that such interpolations, coined *Semi-Ensembles*, can achieve a graceful accuracy-compactness tradeoff. This requires a careful construction of the joint parameter space, i.e., selecting the shared bases from each model. We find that two algorithmic choices play a key role:

- *Sharing parameters for neurons with high activation correlation*: We find that using parameters that correspond to neurons with highly correlated activation as the shared basis is an effective strategy. Such criterion is already being actively used to identify how to permute the neurons before interpolation, and we find it useful for our purpose as well [17]. Intuitively, this choice can be interpreted as interpolating only highly similar features of endpoint models, and preserving the unique features of each model without any damage (thereby giving a multi-view model [2]).
- *Less over-parameterization for later layers:* We empirically find that having a larger number of shared bases (thus less over-parameterized) on the later layers is essential to achieve a good compactness-accuracy tradeoff. This observation is well aligned with the pruning literature that later layers tend to be more compressible [9, 20], and is in contrast with competing approach that avoids interpolating later layers [27].

Our experiments show that semi-ensembles achieve a better accuracy-compactness tradeoff than competing frameworks for over-parameterized model merging, e.g., Stoica et al. [27]. Intriguingly,



(a) CIFAR-10 - Balanced split.

(b) CIFAR-10 - 80/20 split.

Figure 2: **Distribution of correlation between matched activations.** We measure the correlation in neural activation patterns between corresponding neurons in two distinct ResNet20x4 architectures, independently trained on the split CIFAR-10 dataset. We observe that many of the activations are close to zero, especially when models are trained on highly heterogeneous data splits.

for the case where the datasets for endpoint models are extremely heterogeneous, the semi-ensemble even outperforms the full model ensemble with a slightly less number of parameters. We also show that tuning the layerwise over-parameterization ratio plays a critical role in achieving a good tradeoff, making a performance difference up to $\sim 3\%$ in the case of CIFAR-10 classification.

2 Preliminaries

We are interested in interpolating two different neural networks of same depth, model A and B. For a simple presentation, we assume that both *endpoint* models are multi-layered perceptrons, but all results in this section can be easily extended to any architectures, e.g., convolutional neural networks. The weight parameters of the l-th layer of model A is denoted by $\mathbf{W}_l^{\mathsf{A}} = \begin{bmatrix}\mathbf{w}_{l,1}^{\mathsf{A}} & \dots & \mathbf{w}_{l,n_l}^{\mathsf{A}}\end{bmatrix}^\top$ where each vector $\mathbf{w}_{l,i}^{\mathsf{A}}$ denotes the weights connected to the i-th output neuron (among n_l) of the layer. The corresponding activations at the layer output, for some sample given, are denoted by the n_l-dimensional vector $\mathbf{x}_l^{\mathsf{A}} = \begin{bmatrix}x_{l,1}^{\mathsf{A}} & \dots & x_{l,n_l}^{\mathsf{A}}\end{bmatrix}$.

Permuting the neurons. One useful property of neural networks is that the output of the model does not change as we permute the order of the neurons (i.e., *permutation-invariant*). More precisely, suppose that we have some $n_l \times n_l$ permutation matrix $\mathbf{P}$. Then, suppose that we construct a new model $\tilde{\mathsf{A}}$ by permuting only two layers of model A and leaving all other layers identical. In particular, we permute the l-th layer and $(l+1)$-layer of the model with $\mathbf{P}$, so that

$$\mathbf{W}_l^{\tilde{\mathsf{A}}} = \mathbf{P}\mathbf{W}_l^{\mathsf{A}}, \qquad \mathbf{W}_{l+1}^{\tilde{\mathsf{A}}} = \mathbf{W}_{l+1}^{\mathsf{A}}\mathbf{P}^\top. \tag{1}$$

Then, one can see that the model $\tilde{\mathsf{A}}$ is functionally identical to the model A.

Mode connectivity, modulo permutation. Many works have reported an interesting phenomenon called the *linear mode connectivity* of neural networks [11, 10, 30]: if two models share a substantial amount of training history, one can generate a new model (call C) that has a low training loss by linearly interpolating the parameters of two models that have the same size (call A, B). Formally, the weight parameters of the l-th layer of the interpolated network can be written as

$$\mathbf{W}_l^{\mathsf{C}} = \alpha \cdot \mathbf{W}_l^{\mathsf{A}} + (1-\alpha) \cdot \mathbf{W}_l^{\mathsf{B}}, \qquad \alpha \in [0,1]. \tag{2}$$

This operation can be viewed as *merging the neurons* of endpoint networks one-to-one, where the neurons to match are determined by the original ordering of each neuron.

Interestingly, several findings suggest that the linear mode connectivity may also hold for the endpoint models that have been trained independently from each other, once we are allowed to permute the neurons of an endpoint model to *match* the counterpart neurons before interpolation [21, 26]. The

success of such algorithms is largely determined by how well such algorithms can identify highly similar pairs of neurons from endpoint models.

Matching algorithms. Typically, model interpolation literature formulates the neuron matching as an optimization problem; the goal is to maximize the pairwise similarity of the matched neuron pairs in each layer, using various notions of similarity. A popular approach is to use the (Pearson) correlation between the activations of the neurons as a similarity measure, where the activations are computed based on some batch of training data [1, 8, 17]. More formally, the layerwise neuron matching for l-th layer is to solve the maximization problem

$$\max_{\mathbf{P}_l} \sum_{i=1}^{n_l} \texttt{corr}\left(\mathbf{x}_{l,i}^{\mathsf{A}}, \mathbf{x}_{l,\mathbf{P}_l(i)}^{\mathsf{B}}\right), \tag{3}$$

where $\texttt{corr}(\cdot,\cdot)$ denotes the Pearson correlation and $\mathbf{P}_l(i)$ is the index where the i-th index is mapped to by the permutation matrix $\mathbf{P}_l$. The maximization can be solved by standard combinatorial optimization algorithms, e.g., Hungarian algorithm. Whenever the solutions for eq. (3) have been found, model merging algorithms interpolate the weight parameters as

$$\mathbf{W}_l^{\mathsf{C}_\alpha} = \alpha \cdot \mathbf{W}_l^{\mathsf{A}} + (1-\alpha) \cdot \mathbf{P}_l \mathbf{W}_l^{\mathsf{B}} \mathbf{P}_{l-1}. \tag{4}$$

After interpolating the weights (and also corresponding biases), one can additionally perform simple renormalization steps, which can substantially boost the performance of the merged models [17].

2.1 A motivating observation: Are all matched neurons sufficiently correlated?

As shown in many prior works, interpolating two models without neuron matching results in a large performance drop, because randomly matched neurons would lose most of their information after interpolation [1, 17]. In other words, interpolating less-correlated neurons with low similarity may cause performance degradation. Motivated by this, we first question whether all matched neurons are sufficiently activation-correlated with each other. The concept of *multi-view* provides ground for our suspicion [2]. Allen-Zhu & Li [2] claims it as a mechanism of model ensemble, that independently trained models can capture their unique features. In this paper, we consider a model merging when there exist unique views (i.e. multi-view) between model endpoints.

Observation. In Figure 2, we observe that some neuron pairs show near-zero correlations when trained on separate data splits. In other words, no matter how well we match the neurons, some neurons may need to be merged with uncorrelated counterpart neurons. Comparing Figure 2a and 2b, we can also observe heterogeneous training data split leads to an increased number of neuron pairs with small correlations. It implies that we need a new type of model merging algorithm which considers inevitable low-correlated neuron pairs, i.e. unique view of each model.

3 Semi-Ensemble

We now describe Semi-Ensemble, an algorithm to generate over-parameterized interpolations of the models by partially interpolating the endpoint model parameters. Intuitively, Semi-Ensemble only interpolates the weights of highly-correlated neurons and leaves less correlated features untouched.

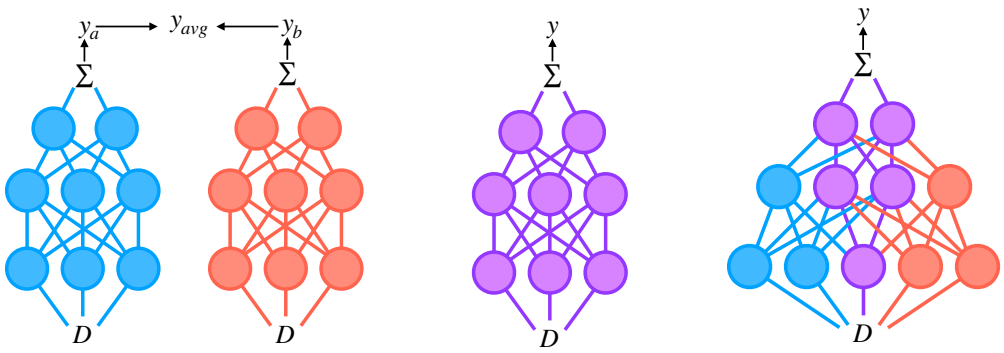
3.1 Layerwise operation

We interpolate the neural network layers one-by-one, sequentially from the earliest layer to the latest. The layerwise interpolation for the l-th layer can be described as follows.

Without loss of generality, we assume that the weight matrices of the endpoint models have been already permuted. Furthermore, we assume that the output neurons have been sorted in the descending order of the activation correlations, i.e.,

$$\texttt{corr}(\mathbf{x}_{l,1}^{\mathsf{A}}, \mathbf{x}_{l,1}^{\mathsf{B}}) \geq \texttt{corr}(\mathbf{x}_{l,2}^{\mathsf{A}}, \mathbf{x}_{l,2}^{\mathsf{B}}) \geq \cdots \geq \texttt{corr}(\mathbf{x}_{l,n_l}^{\mathsf{A}}, \mathbf{x}_{l,n_l}^{\mathsf{B}}). \tag{5}$$

We interpolate only the first k_l output neurons from each model, and keep the remaining $n_l - k_l$ neurons from each model untouched. In other words, the interpolated weight matrix may have the output dimension equal to $2n_l - k_l$, with its first k_l neurons being interpolated neurons and the latter $2n_l - 2k_l$ neurons being uninterpolated neurons.



(a) Model ensemble (b) Merged model (c) Semi-Ensemble (ours)

Figure 3: **A visual comparison of ensemble, model merging, and semi-ensemble.** Three different methods to utilize the knowledge of endpoint models. The neurons and parameters from each endpoint model are colored blue (or red), and the merged neurons and parameters are colored purple.

Step#1. Over-parameterization. Before merging the output neurons, we *over-parameterize* each endpoint model by copy-and-pasting the missing weight values from the counterpart model. To be precise, note that there are several "zero" entries in both weight matrices $\mathbf{W}_l^{\mathsf{A}} = \begin{bmatrix}\mathbf{w}_{l,1}^{\mathsf{A}} & \dots & \mathbf{w}_{l,n_l}^{\mathsf{A}}\end{bmatrix}^\top$ and $\mathbf{W}_l^{\mathsf{B}} = \begin{bmatrix}\mathbf{w}_{l,1}^{\mathsf{B}} & \dots & \mathbf{w}_{l,n_l}^{\mathsf{B}}\end{bmatrix}^\top$, because there are some input neurons of the layer that came exclusively from a single endpoint model without any merging (either from model A or B). If the i-th input neuron is exclusively from A, the weight from the i-th input neuron to the j-th output neuron on the model B, i.e., $w_{l,i\to j}^{\mathsf{B}}$ is destined to be zero. To such zero weights, we copy the value from the other model so that $w_{l,i\to j}^{\mathsf{B}} = w_{l,i\to j}^{\mathsf{A}}$. This operation can be seen as finding the over-parameterized version of an endpoint model that is closest to the hyperplane given by the other model (see Figure 1b).

Step#2. Interpolation. We now interpolate the first k_l output neurons of the endpoint models. In other words, we generate $\mathbf{W}_l^{\mathsf{C}} = \begin{bmatrix}\mathbf{w}_{l,1}^{\mathsf{C}} & \cdots & \mathbf{w}_{l,2n_l-k_l}^{\mathsf{C}}\end{bmatrix}^\top$ with the formula:

$$\mathbf{w}_{l,i}^{\mathsf{C}_\alpha} = \begin{cases} \alpha \cdot \mathbf{w}_{l,i}^{\mathsf{A}} + (1-\alpha)\cdot \mathbf{w}_{l,i}^{\mathsf{B}} & \cdots \quad i \in \{1,\dots,k_l\}, \\ \mathbf{w}_{l,i}^{\mathsf{A}} & \cdots \quad i \in \{k_l+1,\dots,n_l\}, \\ \mathbf{w}_{l,i+k_l-n_l}^{\mathsf{B}} & \cdots \quad i \in \{n_l+1,\dots,2n_l-k_l\}. \end{cases} \tag{6}$$

Although width of lth layer is extended from n_l to $2n_l - k_l$ in Equation (6), the extended layer represents a structuredly pruned layer. The connections such as those from the blue to the red neurons, which have zero weights, can be disregarded easily during computation. It means that size of layer does not scale quadratically to the width extended. After interpolation, we implement a series of renormalization procedures on the Batch Normalization layers [15], as delineated in the REPAIR[17].

Note. To interpolate the last layer, we do not perform the over-parameterization step, and only perform the interpolation step; this is equivalent to taking an ensemble of two submodels.

3.2 Tuning the layerwise k_l

One benefit of our framework is that we can flexibly choose the number of neurons to be merged in each layer. In our experiments in Section 4.3, we perform a grid-search to select the best layerwise merging ratio (or layerwise degree of over-parameterization) to find the best performing model given the global constraint on the number of parameters. We find that, in general, having a larger layerwise merging ratio for the latter layers is better for the performance.

4 Experiment

In this section, we compare the performance of the Semi-Ensemble with model interpolation, model ensemble, and a recently proposed partial model interpolation algorithm [27]. We focus on the evaluation on the image classification task, following the prior works.

4.1 Experimental Setup

Datasets. We mainly evaluate on various splits of the CIFAR-10 image classification dataset.

- **CIFAR-10 - Balanced** is a basic benchmark task whose class distribution is balanced for each training data split. This setting is the simplest benchmark for both conventional model ensemble and model merging algorithms. Each model can learn different 2500 training images per class. For this benchmark, we only report test accuracy on balanced test set.
- **CIFAR-10 - 80/20 split** is a CIFAR10 split dataset used in previous model merging tasks [1, 17]. With randomly selected five classes, one model can access 80% of training data for selected classes and 20% of training data for not-selected classes. Counterpart model can access to complement of them.
- **CIFAR-10 - Disjoint** is a CIFAR10 split dataset used as main benchmark in ZipIt! [27]. One model can access to full training data on randomly selected five classes and the other model can access training data with other five classes.

For 80/20 split and disjoint benchmark, we name the training task used for training model X as Task X. Classes for each task are randomly selected. Accuracy without additional tag is a test accuracy for balanced test set. During calculating correlation and resetting batch normalization [15] statistics, union of both training dataset is used.

Models. We mainly use ResNet20x4, which is a variant of the standard ResNet20 model, characterized by having its channels expanded four times. We use the standard linear classification layer, instead of a CLIP-based classifiers.

Baselines. We compare the performance of merged model with two available endpoints of our method and baseline model merging algorithm.

- **REPAIR** [17] is a model merging algorithm which aims for one-to-one correspondence matching of all neurons across model. In our algorithm, REPAIR is a special case when k in Equation (6) is zero for all layers, which fully merge all neurons.
- **Ensemble** is a deep learning algorithm, which inferences multiple models to boost final performance. Prediction averaging is one of the simplest ensemble algorithms which naïvely averages logits from ingredient models. When all k in Equation (6) is equal to n_l, two models are attached into single model but do not share any neurons instead of the first convolutional and last classifier layer. In our experiment, *Ensemble* directs prediction averaging.
- **ZipIt!** [27] is a recently proposed partial model interpolation algorithm that can merge only a fraction of early layers instead of all layers (called *partial zipping*). We compare with various degrees of partial zipping.

Hyperparameters. To train single ResNet20x4 model, we use SGD optimizer with momentum 0.9, weight decay $1\mathrm{e}{-4}$ and cosine learning rate schedule starting from 0.4. During 200 epochs training, batch size is set to 500 and `flip+translate+cutout` is adapted for training data augmentation. Other details for specific experiments are noted in each section.

Seeds. All results are averaged over three different seeds.

4.2 Main Results

The main results are reported in Table 1. We can summarize observations as:

- Semi-Ensemble achieves higher test accuracy compared with ZipIt! [27] for all benchmark datasets and various number of parameters.
- Our method can achieve flexible interpolated model size (e.g. $\times 1.38$), which cannot be achieved by ZipIt! [27] algorithm.
- In CIFAR-10 - Disjoint benchmark, our method with parameter multiplication $\times 1.95$ results 79.19% test accuracy, which is even better than test accuracy of ensemble, 78.02%.

4.3 Tuning layer-wise merging ratios

We now test one core benefit of our method: tuning layer-wise merging ratio, which is explained in Section 3.2. For simplicity, we test effect of tuning merging ratio on our main setting, CIFAR-10-Balanced task.

Table 1: **Comparing performance of Semi-Ensemble and other model merging methods.** Semi-ensemble is compared with other baselines on ResNet20x4 with linear classification layers trained with various CIFAR-10 dataset. † is a result that Semi-Ensemble outperforms model ensemble.

Method	Num Params (multiplied)	Balanced	80/20 Split			Disjoint		
		Acc. (%)	Acc. (%)	Per-task		Acc. (%)	Per-task	
				Task A	Task B		Task A	Task B
Model A	×1.0	92.01	89.14	93.12	85.17	48.47	96.94	0.0
Model B	×1.0	92.05	87.18	85.06	89.30	47.53	0.0	95.05
REPAIR	×1.0	85.74	84.92	85.04	84.83	52.78	52.78	52.78
ZipIt!(20/20)	×1.0	83.94	84.04	82.57	85.74	45.60	46.84	44.45
ZipIt!(13/20)	×1.78	89.72	84.71	85.82	84.28	62.81	63.82	61.94
ZipIt!(7/20)	×1.95	91.53	87.68	88.73	86.19	66.81	68.09	65.52
Semi-Ensemble (Ours)	×1.38	89.92	84.91	84.05	85.77	63.52	64.38	62.66
Semi-Ensemble (Ours)	×1.77	92.88	86.59	85.01	88.17	76.94	76.09	77.79
Semi-Ensemble (Ours)	×1.95	93.10	88.38	88.36	88.39	79.19†	79.17	79.22
Ensemble	×2.0	93.48	91.44	93.19	90.02	78.02	76.29	79.24

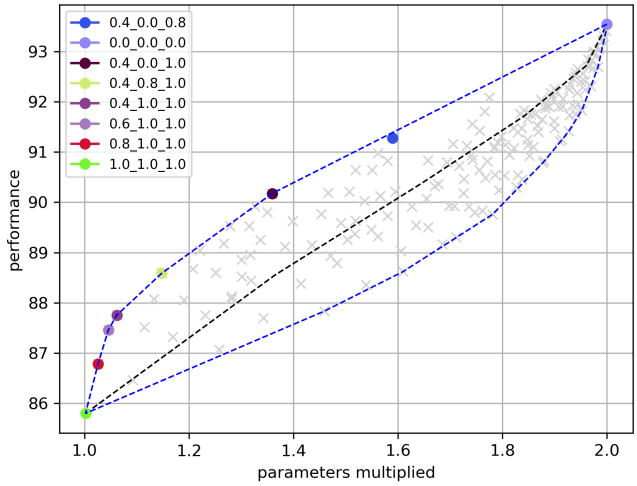


Figure 4: **Performance of interpolated model with various layer-wise merging ratios on CIFAR-10-Balanced.** Legend represents block-wise merging ratio in order. For example, in 0.4_0.8_1.0, 40% of neurons in first block are merged and 100% (i.e. all) of neurons in last block are fully merged. Blue dashed line is a convex hull of hyperparameter search points to show trend of best accuracy on given merging ratio. Black dashed line is performance of the merged model with uniform merging ratio among all layers.

Search space. To search for optimal layer-wise merging strategy efficiently, we evaluate block-by-block grid search where each block is a list of convolutional layers with same number of output channels. That means, ResNet20x4 can be split into 3 blocks in our experiment. This criterion for splitting model is referenced from results in Stoica et al. [27]. For each block, merging ratio is searched in range $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$.

4.3.1 Result

Figure 4 depicts the performance of merged model with various layer-wise merging ratios. Each point represents the outcome of a merged model, identified through a hyperparameter tuning process. Compared with the performance of a uniformly merged model (black dashed line), the performance of the model with well-tuned merging ratio is much higher. Given the global constraint on the number of parameters, we find that the models that merge more neurons in the latter layers tend to work better.

Table 2: **Ablation study for Semi-Ensemble.** Table 2a is a table for Section 4.4.1, which shows that merging neuron pairs with high correlation is efficient strategy. Table 2b is a table for Section 4.4.2, which argues that last linear layer should be treated specially as we proposed.

Deep Num params	**Top**	Random	Bot
$\times 1.0$ (REPAIR)	85.74	-	-
$\times 1.38$	**89.92**	89.37	88.93
$\times 1.52$	**90.93**	90.46	90.09
$\times 1.77$	**92.88**	91.91	91.60
$\times 2.0$(Ensemble)	93.48	-	-

(a) **Performance difference when priority of neuron pair differs** in CIFAR-10 - Balanced task. **Top** and **Bot** respectively means neuron pairs with high/low activation correlations are merged for each layer. **Random** means that matched neurons are randomly selected and used for merging.

Deep Num params	**SE**	Naïve	Partial
$\times 1.0$(REPAIR)	85.74	-	-
$\times 1.38$	89.92	86.65	**89.98**
$\times 1.52$	**90.93**	87.53	90.83
$\times 1.77$	**92.88**	89.37	92.21
$\times 2.0$(Ensemble)	93.48	-	-

(b) **Performance difference when last layer merging strategy differs** in CIFAR-10 - Balanced task. **SE** denotes the Semi-Ensemble. **Naïve** is a variant of SE, which does not use Step#1 of Section 3. **Partial** is another variant of SE, which performs Step#1 in the last layer as well.

We note that this result is well-aligned with the pruning literature on convolutional neural networks [9, 20], that the latter layers in the model can be compressed to a higher degree.

4.4 Ablation

Now we do ablation studies of Semi-Ensemble. In this section, we report contribution of two components in our algorithm, merging neurons with high correlation and merging strategy for classifier layer, on final performance. As a base experiment, we fix tuning strategy to be *deep*, an optimal tuning strategy with limited number of total parameters searched in Figure 4, for fair comparison. All experiments are done on CIFAR-10-Balanced task.

4.4.1 Effect of merging neurons with high correlation

Our main criterion to select neurons to be merged is how large the correlation [21] between neuron activations. In Table 2a, we now empirically test how performance differs when criterion to select neurons is reversed (Bot, i.e. neuron pairs with low correlation are merged) or randomly selected (Random). We can observe that, in same layer-wise merging ratio, merging neurons with high correlation (Top) is the most optimal criterion among three options.

4.4.2 Effect of different merging strategies for classifier layer

In Section 3, we specially treat linear classifier layer to mimic logit averaging in ensemble scenario. In Table 2b, we ablate how performance differs when different linear layer merging strategies are adopted. It shows that merging last layer with our proposed approach performs slightly better than merging entire model with partial merging, except for lower number of parameters ($\times 1.38$). One observation we find is that performance gap between *Mix* and *Partial* is amplified when the interpolated model becomes larger (i.e. closer to model ensemble). We can see that this method works as we intended, mimicking output of model ensemble.

5 Related Works

Model ensemble. Model ensemble [6, 3, 4, 19] is a technique to improve deep learning performance by using multiple models at the inference time. Model ensemble are not only effective in the standard scenario where all models are trained on the same data split, but is also powerful when aggregating the models trained on highly heterogeneous datasets with complementary knowledge [34, 33]. Despite its effectiveness, model ensemble comes with inherent limitations, computation and memory requirements during inference. Although techniques for lightweight model ensemble have been developed before [32, 5], it still require resources proportional to the number of models.

Model merging and linear mode connectivity. Model merging is a technique to merge two or more models into single model. According to Model Soup [30], it was shown that performance can be boosted by simply averaging weights of independently trained model endpoints if all model endpoints are located in same error basin [25]. This naïve weight averaging can boost performance without increasing inference cost as model ensemble [6]. High performance of *model merging* is grounded on the idea of mode connectivity [7, 12, 10, 11], the concept that there exists a path of near-constant loss to connect two optimal model endpoints. This naïve weight averaging was widely applied to practical applications earlier while training single model [14, 16] or on federated learning [24, 22]. To average weights from different error basins (e.g. models trained from different initialization), permutation invariance [8] allows models to move to another basin without loss of performance [28]. Git Re-Basin [1] and REPAIR [17] respectively match weights and activations of two models by solving linear assignment problem [18] for optimal permutation.

Model merging with exclusive views. However, it is not always possible to match all neurons from counterpart model. Allen-Zhu & Li [2] explain how model ensemble works by introducing concept *multi-view*, that independently trained models can learn only a few features in the dataset. There are emerging observations that linear path may not exist between models with exclusive feature [23] and naïvely averaging them can lead performance degradation of merged model [31]. One previous work [13] proposed an algorithm, merging models trained for different tasks by maintaining task-specific neurons (i.e. features not learned in counterpart model) unmerged. Recently, as an algorithm to overcome the limitation of one-to-one correspondence matching and get advantage of exclusive features during merging, ZipIt! [27] preserves exclusive features by matching redundant activations within model and even leaves deeper layers as multi-head architecture. Although it can generate merged model that performs well for both tasks, merged model has the complex architecture to be efficiently deployed in various environments. In addition, to be compared with model ensemble methods [6, 3, 4, 19], existing matching-based model merging algorithms [1, 17, 27] still have performance gap or parameter-inefficiency.

6 Conclusion

In this paper, we propose Semi-Ensemble, a simple model merging method which results midpoint of model merging and model ensemble. Motivated by the observation that some matched neurons are not highly similar and can lose information during interpolation, we preserve those neurons with dissimilar counterparts by extending dimension of model and allocating each neuron to new dimension. With proper tuning that other baselines cannot provide, Semi-Ensemble outperforms other model merging baselines in various training data distributions and even outperforms model ensemble in label-split training datasets.

Limitations. One limitation of Semi-Ensemble is the heuristic nature of tuning layer-wise merging ratio. It requires an additional computation to find optimal merging ratio, which requires non-negligible time to be adapted to iterative process such as a federated learning system. Another limitations of presented Semi-Ensemble is the increase in size during the model merging process, which consequently complicates the sequential merging of more than two models. We leave the study of merging more than two models simultaneously for the future.

References

[1] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-basin: Merging models modulo permutation symmetries. In International Conference on Learning Representations, 2023.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In International Conference on Learning Representations, 2023.

[3] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36:105–139, 1999.

[4] Leo Breiman. Bagging predictors. Machine Learning, 24:123–140, 1996.

[5] George DC Cavalcanti, Luiz S Oliveira, Thiago JM Moura, and Guilherme V Carvalho. Combining diversity measures for ensemble pruning. Pattern Recognition Letters, 74:38–45, 2016.

[6] Thomas G Dietterich. Ensemble methods in machine learning. In International workshop on multiple classifier systems, pp. 1–15. Springer, 2000.

[7] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In Proceedings of the International Conference on Machine Learning, pp. 1309–1318. PMLR, 2018.

[8] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In International Conference on Learning Representations, 2022.

[9] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In Proceedings of the International Conference on Machine Learning, pp. 2943–2952. PMLR, 2020.

[10] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Proceedings of the International Conference on Machine Learning, pp. 3259–3269. PMLR, 2020.

[11] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. arXiv preprint arXiv:1611.01540, 2016.

[12] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In Advances in Neural Information Processing Systems, volume 31, 2018.

[13] Xiaoxi He, Zimu Zhou, and Lothar Thiele. Multi-task zipping via layer-wise neuron sharing. Advances in Neural Information Processing Systems, 31, 2018.

[14] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In International Conference on Learning Representations, 2017.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, pp. 448–456. pmlr, 2015.

[16] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

[17] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing Permuted Activations for Interpolation Repair. In International Conference on Learning Representations, 2023.

[18] Harold W Kuhn. The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2):83–97, 1955.

[19] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Advances in Neural Information Processing Systems, volume 30, 2017.

[20] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In International Conference on Learning Representations, 2016.

[21] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? arXiv preprint arXiv:1511.07543, 2015.

[22] 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 Proceedings of the International Conference on Machine Learning, pp. 13857–13869. PMLR, 2022.

[23] Ekdeep Singh Lubana, Eric J Bigelow, Robert P Dick, David Krueger, and Hidenori Tanaka. Mechanistic mode connectivity. In Proceedings of the International Conference on Machine Learning, pp. 22965–23004. PMLR, 2023.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 1273–1282. PMLR, 2017.

[25] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In Advances in Neural Information Processing Systems, volume 33, pp. 512–523, 2020.

[26] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In Advances in Neural Information Processing Systems, volume 33, pp. 22045–22055, 2020.

[27] George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. ZipIt! Merging models from different tasks without training. arXiv preprint 2305.03053, 2023.

[28] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. In Advances in Neural Information Processing Systems, volume 33, pp. 15300–15311, 2020.

[29] Honyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In International Conference on Learning Representations, 2020.

[30] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Proceedings of the International Conference on Machine Learning, pp. 23965–23998. PMLR, 2022.

[31] Masanori Yamada, Tomoya Yamashita, Shin'ya Yamaguchi, and Daiki Chijiwa. Revisiting permutation symmetry for merging models between different datasets. arXiv preprint 2306.05641, 2023.

[32] Yi Zhang, Samuel Burer, W Nick Street, Kristin P Bennett, and Emilio Parrado-Hernández. Ensemble pruning via semi-definite programming. Journal of Machine Learning Research, 7 (7), 2006.

[33] Yifan Zhang, Bryan Hooi, Lanqing Hong, and Jiashi Feng. Self-supervised aggregation of diverse experts for test-agnostic long-tailed recognition. In Advances in Neural Information Processing Systems, volume 35, pp. 34077–34090, 2022.

[34] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9719–9728, 2020.