Harmony in Diversity: Merging Neural Networks with Canonical Correlation Analysis

Stefan Horoi [1 2] Albert Manuel Orozco Camacho [3 2] Eugene Belilovsky [3 2] Guy Wolf [1 2]

Abstract

Combining the predictions of multiple trained models through ensembling is generally a good way to improve accuracy by leveraging the different learned features of the models, however it comes with high computational and storage costs. Model fusion, the act of merging multiple models into one by combining their parameters reduces these costs but doesn't work as well in practice. Indeed, neural network loss landscapes are high-dimensional and non-convex and the minima found through learning are typically separated by high loss barriers. Numerous recent works have been focused on finding permutations matching one network features to the features of a second one, lowering the loss barrier on the linear path between them in parameter space. However, permutations are restrictive since they assume a one-to-one mapping between the different models' neurons exists. We propose a new model merging algorithm, CCA Merge, which is based on Canonical Correlation Analysis and aims to maximize the correlations between linear combinations of the model features. We show that our alignment method leads to better performances than past methods when averaging models trained on the same, or differing data splits. We also extend this analysis into the harder setting where more than 2 models are merged, and we find that CCA Merge works significantly better than past methods. [1]

1. Introduction

A classical idea for improving the predictive performance and robustness of machine learning models is to use multiple trained models simultaneously. Each model might learn to extract different, complementary pieces of information from the input data, and combining the models would take advantage of this complementarity; thus benefiting the final performance (Ho, 1995). One of the simplest ways to implement this idea is to use ensembles, where multiple models are trained on a given task and their outputs are combined through averaging or majority vote at inference (Ho, 1995; Lobacheva et al., 2020). While this method yields good results, it comes with the disadvantages of having to store in memory the parameters of multiple models and having to run each of them individually at inference time, resulting in high storage and computational costs, particularly in the case of neural networks (NNs). Another way of leveraging multiple models to improve predictive performance is to combine the different sets of parameters into a single model. This is typically done through averaging or interpolation

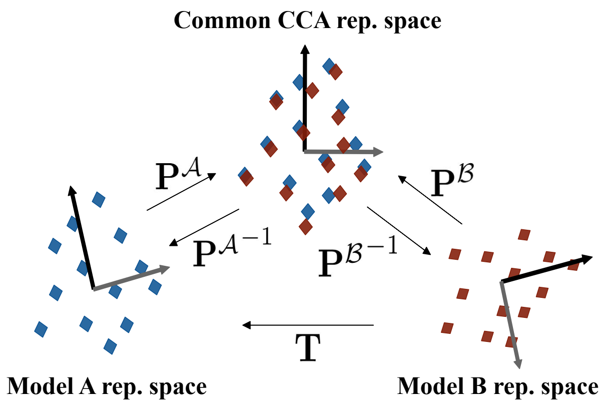
Common CCA rep. space
$\mathbf{P}^{\mathcal{A}}$
$\mathbf{P}^{\mathcal{B}}$
$\mathbf{P}^{\mathcal{A}^{-1}}$
$\mathbf{P}^{\mathcal{B}^{-1}}$
$\mathbf{T}$
Model A rep. space
Model B rep. space

Figure 1: **Visual representation of using CCA Merge to align two models.** Canonical Correlation Analysis is used to find a common representation space where orthogonal linear combinations of the features (neurons) from $\mathcal{A}$ and $\mathcal{B}$ are maximally correlated. The linear transformation $\mathbf{P}^{\mathcal{A}}$ (resp. $\mathbf{P}^{\mathcal{B}}$) and its inverse can be used to go from the representation space of model $\mathcal{A}$ (resp. model $\mathcal{B}$) to this common representation space and back. By applying $\mathbf{P}^{\mathcal{B}}$ first and then $\mathbf{P}^{\mathcal{A}^{-1}}$ we can align the representations of model $\mathcal{B}$ to those of model $\mathcal{A}$. Applying the same transformation directly to the parameters of model $\mathcal{B}$ effectively aligns the two models, thus allowing their merging.

[1]Université de Montréal [2]Mila - Quebec AI Institute [3]Concordia University. Correspondence to: Stefan Horoi <stefan.horoi@umontreal.ca>, Guy Wolf <guy.wolf@umontreal.ca>, Eugene Belilovsky <eugene.belilovsky@concordia.ca>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Our code is publicly available at `https://github.com/shoroi/align-n-merge`

in the parameter space of the models. After this "fusion" only a single model remains which will be used at inference time; therefore, the storage and computational costs are minimized, being the same as for a single model. The downside of such model fusion methods is that existing methods are not robust and typically do not perform as well in practice as ensembling (Stoica et al., 2024). Neural networks are highly over-parameterized for the task they solve (Arpit et al., 2017) and their loss landscapes are high-dimensional and non-convex objects which are still somewhat poorly understood despite many recent works shedding light on some of their characteristics (Goodfellow & Vinyals, 2015; Keskar et al., 2017; Li et al., 2018; Horoi et al., 2022). Multiple good local minima can be found for a given model and task but these minima are most often separated by regions of high loss (Frankle et al., 2020). Therefore, combining the parameters from multiple trained models without falling into one of these high-loss regions and destroying the features learned during training is a hard task and constitutes an active area of research.

Previous works established empirically that any two minima in an NN parameter space found through SGD and its variants are linked by a non-linear low-loss path (Garipov et al., 2018; Draxler et al., 2018). The term *mode connectivity* describes this phenomenon. However, to find this low-loss path between two minima one needs to run a computationally expensive optimization algorithm. As such, model fusion based on nonlinear mode connectivity has not been explored. On the other hand, *linear mode connectivity* which describes two optima connected by a *linear* low loss path in the parameter space (Frankle et al., 2020), provides a straightforward way of combining these models. Indeed, if the loss remains low on the linear path from one model to the other, merging the two models is as simple as averaging or linearly interpolating their parameters. This has emerged as a simple, yet powerful way to compare NN minima. However, this phenomenon is very rare in practice and is not guaranteed even for networks with the same initializations (Frankle et al., 2020).

One reason linear mode connectivity is hard to obtain is due to the well-known NN invariance to permutations. Indeed, it is possible to permute the neurons of an NN layer without changing the actual function learned by the model as long as the connections/weights to the subsequent layer are permuted consistently. Therefore, it is possible to have the same features learned at every layer of two different NN models and them not be linearly mode-connected if the order of the features differs from one network to the other. Using this invariance as justification, Entezari et al. (2022) conjectured that most SGD solutions can be permuted in such a way that they are linearly mode connected to most other SGD solutions and presented empirical support for this conjecture. Many works in recent years have provided algorithms for finding permutations that successfully render pairs of SGD solutions linearly mode connected, or at least significantly lower the loss barrier on the linear path between these solutions, further supporting this conjecture (Tatro et al., 2020; Singh & Jaggi, 2020; Peña et al., 2023; Ainsworth et al., 2023).

While these algorithms and the found transformations have been successful in lowering the loss barrier when interpolating between pairs of SGD solutions most of them do not consider the possibility that perhaps other linear transformations, besides permutations, would provide an even better matching of NN weights. While the permutation conjecture is enticing given its simplicity and NNs' invariance to permutations, there is nothing inherently stopping NNs from distributing computations that are done by one neuron in a model to be done by multiple neurons in another model. Permutations would fail to capture this relationship since it is not a one-to-one mapping between features. Furthermore, the focus of recent works has been mainly on merging pairs of models, and merging multiple models has received limited study. However, if a similar function is learned by networks trained on the same task then it should be possible to extract the commonly learned features from not only two but also a larger population of models. Model merging algorithms should therefore be able to find these features and the relationship between them and then merge many models without negatively affecting performance.

Contributions In this work we introduce CCA Merge, a more flexible way of merging models based on maximizing the correlation between linear combinations of neurons. Furthermore we focus on the difficult setting of merging not only two but also *multiple* models which were *fully trained from random initializations*. Our main contributions are threefold:

- We propose a new model merging method based on Canonical Correlation Analysis (Sec. 3) which we will refer to as "CCA Merge". This method is more flexible than past, permutation-based methods and therefore makes better use of the correlation information between neurons (Sec. 4.2).
- We compare CCA Merge to past works and find that it yields better performing merged models across a variety of architectures and datasets. This is true in both settings where the models were trained on the same data (Sec. 4.3) or on disjoint splits of the data (Sec. 4.5).
- We take on the difficult problem of aligning features from multiple models and then merging them. We find that CCA Merge is significantly better at finding the common learned features from populations of NNs and aligning them, leading to lesser accuracy drops as the number of models being merged increases (Sec. 4.4).

2. Related Work

Mode connectivity Freeman & Bruna (2017) proved theoretically that one-layered ReLU neural networks have asymptotically connected level sets. Garipov et al. (2018) and Draxler et al. (2018) explore these ideas empirically and introduce the concept of *mode connectivity* to describe ANN minima that are connected by nonlinear paths in parameter space along which the loss remains low. Frankle et al. (2020) introduced the concept of *linear mode connectivity* describing the scenario in which two ANN minima are connected by a *linear* low loss path in parameter space.

Model merging through alignment More recently, Entezari et al. (2022) have conjectured that "Most SGD solutions belong to a set $\mathcal{S}$ whose elements can be permuted in such a way that there is no barrier on the linear interpolation between any two permuted elements in $\mathcal{S}$" or in other words most SGD solutions are linearly mode connected provided the right permutation is applied to align the two solutions. Many recent works seem to support this conjecture, proposing methods for finding the "right" permutations (Tatro et al., 2020; Singh & Jaggi, 2020; Peña et al., 2023; Ainsworth et al., 2023).

"Easy" settings for model averaging Linear mode connectivity is hard to achieve in deep learning models. Frankle et al. (2020) established that even models trained on the same dataset with the same learning procedure and even the same initialization might not be linearly mode connected if they have different data orders/augmentations. It seems that only models that are already very close in parameter space can be directly combined through linear interpolation. This is the case for snapshots of a model taken at different points during its training trajectory (Garipov et al., 2018; Izmailov et al., 2018) or multiple fine-tuned models with the same pre-trained initialization (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023). This latter setting is the one typically considered in NLP research. Another setting that is worth mentioning here is the "federated learning" inspired one where models are merged every couple of epochs during training (McMahan et al., 2017). The common starting point in parameter space and the small number of training iterations before merging make LMC easier to attain. Model fusion has been very successful in these settings where aligning the models prior to merging is not required.

We emphasize that these settings are different from ours in which we aim to merge *fully trained models* with different parameter initializations and SGD noise (data order and augmentations).

Merging multiple models Merging more than two models has only been explored thoroughly in the "easy" settings described above. For example, Wortsman et al. (2022) average models fine-tuned with different hyperparameter configurations and find that this improves accuracy and robustness. Jolicoeur-Martineau et al. (2024) average the weights of a population of neural networks multiple times during training, leading to performance gains. On the other hand, works that have focused on providing feature alignment methods to be able to merge models in settings in which LMC is not trivial have mainly done so for 2 models at the time (Singh & Jaggi, 2020; Ainsworth et al., 2023; Peña et al., 2023; Jordan et al., 2023). An exception to this is Git Re-Basin (Ainsworth et al., 2023) which proposes a "Merge Many" algorithm for merging a set of multiple models by successively aligning each model to the average of all the other models. However, results obtained with this method, which they use to merge up to 32 models, only concern the very simple set-up of MLPs trained on MNIST. Singh & Jaggi (2020) also consider merging multiple models but either in a similarly simple set-up, i.e. 4 MLPs trained on MNIST, or they fine-tune the resulting model after merging up to 8 VGG11 models trained on CIFAR100. We extend this line of work to more challenging settings, using more complex model architectures, we report the merged models accuracies directly without fine-tuning and make this a key focus in our work.

Model merging beyond permutations We note that the two model merging methods based on optimal transport (Singh & Jaggi, 2020; Peña et al., 2023) can also align models beyond permutations. However, in Singh & Jaggi (2020) this only happens when the two models being merged have different numbers of neurons at each layer. When the models have the same number of neurons the alignment matrix found by their method is a permutation, as such the majority of their results are with permutations. The method proposed by Peña et al. (2023) isn't constrained to finding binary permutation matrices but binarity is still encouraged through the addition of an entropy regularizer. Furthermore, our CCA based method is different in nature from both of these since it is not inspired by optimal transport theory.

CCA in deep learning In the general context of deep learning, Canonical Correlation Analysis has been used to align and compare learned representations in deep learning models (Raghu et al., 2017; Morcos et al., 2018; Gotmare et al., 2019), a task which is similar to the feature matching conducted by model merging algorithms. These past works serve as great motivation for the present paper.

3. Using CCA to Merge Models

3.1. Merging Models: Problem Definition

Let $\mathcal{M}$ denote a standard multilayer perceptron (MLP) and layer $L_i \in \mathcal{M}$ denote a linear layer of that model with a σ = ReLU activation function, weights $\mathbf{W}_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and bias $\mathbf{b}_i \in \mathbb{R}^{n_i}$. Its input is the vector of embeddings from the previous layer $\mathbf{x}_{i-1} \in \mathbb{R}^{n_{i-1}}$ and its output can be described as:

$$\mathbf{x}_i = \sigma(\mathbf{W}_i \cdot \mathbf{x}_{i-1} + \mathbf{b}_i)$$

We use " $\cdot$ " to denote standard matrix multiplication.

Problem statement Now consider two deep learning models $\mathcal{A}$ and $\mathcal{B}$ with the same architecture. We are interested in the problem of merging the parameters from models $\mathcal{A}$ and $\mathcal{B}$ in a layer-by-layer fashion. As mentioned in Sec. 2, simply interpolating the models parameters typically doesn't work when the models are trained from scratch from different initializations. We therefore need to *align* the two models' features before averaging them. We use the term "feature" in our work to refer to the individual outputs of a hidden layer neuron within a neural network. We sometimes also use the term "neuron" to refer to its learned feature or, vice-versa, we might use the term "feature" to refer to a neuron and its parameters. Mathematically, we are looking for linear transformations $\mathbf{T}_i \in \mathbb{R}^{n_i \times n_i}$ which can be applied at the output level of model $\mathcal{B}$ layer i parameters to maximize the alignment with model $\mathcal{A}$'s parameters and minimize the interpolation error. The output of the transformed layer i of $\mathcal{B}$ is then:

$$\mathbf{x}_i^{\mathcal{B}} = \sigma(\mathbf{T}_i \cdot \mathbf{W}_i^{\mathcal{B}} \cdot \mathbf{x}_{i-1}^{\mathcal{B}} + \mathbf{T}_i \cdot \mathbf{b}_i^{\mathcal{B}})$$

We therefore also need to apply the inverse of $\mathbf{T}_i$ at the input level of the following layer's weights to keep the flow of information consistent inside a given model:

$$\mathbf{x}_{i+1}^{\mathcal{B}} = \sigma(\mathbf{W}_{i+1}^{\mathcal{B}} \cdot \mathbf{T}_i^{-1} \cdot \mathbf{x}_i^{\mathcal{B}} + \mathbf{b}_{i+1}^{\mathcal{B}})$$

After finding transformations $\{\mathbf{T}_i\}_{i=1}$ for every merging layer in the network we can merge the two model's parameters at every layer:

$$\mathbf{W}_i = \frac{1}{2}(\mathbf{W}_i^{\mathcal{A}} + \mathbf{T}_i \cdot \mathbf{W}_i^{\mathcal{B}} \cdot \mathbf{T}_{i-1}^{-1}) \tag{1}$$

For the biases we have $\mathbf{b}_i = \frac{1}{2}(\mathbf{b}_i^{\mathcal{A}} + \mathbf{T}_i \cdot \mathbf{b}_i^{\mathcal{B}})$. The linear transformations $\mathbf{T}$ therefore need to be invertible so we can undo their changes at the input level of the next layer and they need to properly align the layers of models $\mathcal{A}$ and $\mathcal{B}$. This problem statement is a generalization of the one considered in past works where transformations $\mathbf{T}$ were constrained to being permutations (Tatro et al., 2020; Entezari et al., 2022; Ainsworth et al., 2023; Jordan et al., 2023).

We note that while artificial neural networks are invariant to permutations in the order of their neurons, this is not the case for general invertible linear transformations. Therefore, after applying the transformations to model $\mathcal{B}$ to align it to $\mathcal{A}$ its functionality might be altered. However, our results (Sec. 4) seem to suggest that the added flexibility of merging linear combinations of features outweighs the possible negative effects of this loss in functionality.

Practical considerations In practice, it is often easier to keep model $\mathcal{A}$ fixed and to find a way to transform model $\mathcal{B}$ such that the average of their weights can yield good performance, as opposed to transforming the parameters of both models. Also, depending on the model architecture, it might not be necessary to compute transformations after each single layer, for example, skip connections preserve the representation space, and the last layers of models are already aligned by the training labels. Therefore we refer to the specific layers in a network where transformations must be computed as "merging layers".

Merging multiple models In the case where multiple models are being merged there is a simple way of extending any method which aligns features between two models to the multiple models scenario. Suppose we have a set of models $\{\mathcal{M}_i\}_{i=1}^n$ which we want to merge. We can pick one of them, say $\mathcal{M}_j$ for $1 \leq j \leq n$, to be the *reference model*. Then we can align the features of every other model in the set to those of the reference model and average the weights. While this "all-to-one" merging approach is quite straightforward it seems to work well in practice.

3.2. CCA Merge: Merging models with CCA

The "best" way to align two layers from two different deep learning models and compute the transformations $\mathbf{T}$ is still an open-question and has been the main differentiating factor between past works (see the baselines presented in Sec. 4.3 and "Model merging through alignment" in Sec. 2).

We propose the use of Canonical Correlation Analysis (CCA) to find the transformations which maximize the correlations between linear combinations of the original features from models $\mathcal{A}$ and $\mathcal{B}$. Let $\mathbf{X}_i^{\mathcal{M}} \in \mathbb{R}^{m \times n_i}$ denote the set of outputs (internal representations or neural activations) of the i-th layer of model $\mathcal{M} \in \{\mathcal{A}, \mathcal{B}\}$ in response to m given input examples. We center $\mathbf{X}_i^{\mathcal{M}}$ so that each column (feature or neuron) has a mean of 0.

CCA finds projection matrices $\mathbf{P}_i^{\mathcal{A}}$ and $\mathbf{P}_i^{\mathcal{B}}$ which bring the neural activations $\mathbf{X}_i^{\mathcal{A}}$ and $\mathbf{X}_i^{\mathcal{B}}$ respectively from their original representation spaces into a new, common, representation space through the multiplications $\mathbf{X}_i^{\mathcal{A}} \cdot \mathbf{P}_i^{\mathcal{A}}$ and $\mathbf{X}_i^{\mathcal{B}} \cdot \mathbf{P}_i^{\mathcal{B}}$. The features of this new representation space are orthogonal linear combinations of the original features of $\mathbf{X}_i^{\mathcal{A}}$ and $\mathbf{X}_i^{\mathcal{B}}$, and they maximize the correlations between the two projected sets of representations.

Once the two projection matrices $\mathbf{P}_i^{\mathcal{A}}$ and $\mathbf{P}_i^{\mathcal{B}}$ aligning $\mathbf{X}_i^{\mathcal{A}}$ and $\mathbf{X}_i^{\mathcal{B}}$ respectively have been found through CCA, we can define the transformation $\mathbf{T}_i = \left(\mathbf{P}_i^{\mathcal{B}} \cdot \mathbf{P}_i^{\mathcal{A}^{-1}}\right)^{\top}$. This transformation can be thought of as first bringing the activations of model $\mathcal{B}$ into the common, maximally correlated space between the two models by multiplying by $\mathbf{P}_i^{\mathcal{B}}$ and then applying the inverse of $\mathbf{P}_i^{\mathcal{A}}$ to go from the common space found by CCA to the embedding space of model $\mathcal{A}$. The transpose here is simply to account for the fact that $\mathbf{T}_i$ multiplies $\mathbf{W}_i$ on the left while the $\mathbf{P}_i^{\mathcal{M}}$'s were described as multiplying $\mathbf{X}_i^{\mathcal{M}}$ on the right, for $\mathcal{M} \in \{\mathcal{A}, \mathcal{B}\}$. The averaging of the parameters of model $\mathcal{A}$ and transformed $\mathcal{B}$ can then be conducted following Eq. 1.

Background on CCA In this section we omit the layer index i since it is implicit. CCA finds the projection matrices $\mathbf{P}^{\mathcal{A}}$ and $\mathbf{P}^{\mathcal{B}}$ by iteratively defining vectors $\mathbf{p}^{\mathcal{A}}$ and $\mathbf{p}^{\mathcal{B}}$ in $\mathbb{R}^n$ such that the projections $\mathbf{X}^{\mathcal{A}} \cdot \mathbf{p}^{\mathcal{A}}$ and $\mathbf{X}^{\mathcal{B}} \cdot \mathbf{p}^{\mathcal{B}}$ have maximal correlation and norm 1.

Let $\mathbf{S}^{\mathcal{AA}} = (\mathbf{X}^{\mathcal{A}})^\top \cdot \mathbf{X}^{\mathcal{A}}$, $\mathbf{S}^{\mathcal{BB}} = (\mathbf{X}^{\mathcal{B}})^\top \cdot \mathbf{X}^{\mathcal{B}}$ and $\mathbf{S}^{\mathcal{AB}} = (\mathbf{X}^{\mathcal{A}})^\top \cdot \mathbf{X}^{\mathcal{B}}$ denote the scatter matrix of $\mathbf{X}^{\mathcal{A}}$, the scatter matrix of $\mathbf{X}^{\mathcal{B}}$ and the cross-scatter matrix of $\mathbf{X}^{\mathcal{A}}$ and $\mathbf{X}^{\mathcal{B}}$ respectively.

$$\begin{aligned}\mathbf{p}^{\mathcal{A}}, \mathbf{p}^{\mathcal{B}} &= \underset{\mathbf{p}^{\mathcal{A}}, \mathbf{p}^{\mathcal{B}}}{\arg\max} (\mathbf{p}^{\mathcal{A}})^\top \mathbf{S}^{\mathcal{AB}} \mathbf{p}^{\mathcal{B}} \\ \text{s.t.} \quad & \|\mathbf{X}^{\mathcal{A}} \cdot \mathbf{p}^{\mathcal{A}}\|^2 = (\mathbf{p}^{\mathcal{A}})^\top \cdot \mathbf{S}^{\mathcal{AA}} \cdot \mathbf{p}^{\mathcal{A}} = 1 \\ & \|\mathbf{X}^{\mathcal{B}} \cdot \mathbf{p}^{\mathcal{B}}\|^2 = (\mathbf{p}^{\mathcal{B}})^\top \cdot \mathbf{S}^{\mathcal{BB}} \cdot \mathbf{p}^{\mathcal{B}} = 1\end{aligned}$$

Since these vectors $\mathbf{p}^{\mathcal{A}}$ and $\mathbf{p}^{\mathcal{B}}$ are defined iteratively they are also required to be orthogonal to the vectors found previously in the metrics defined by $\mathbf{S}^{\mathcal{AA}}$ and $\mathbf{S}^{\mathcal{BB}}$ respectively. Formulating this as an ordinary eigenvalue problem and making it symmetric by a change of variables allows us to find the closed-form solutions as being the vectors in $\mathbf{P}^{\mathcal{A}}$ and $\mathbf{P}^{\mathcal{B}}$ defined by:

$$\begin{aligned}&\mathbf{U}, \mathbf{S}, \mathbf{V}^\top = \text{SVD}((\mathbf{S}^{\mathcal{AA}})^{-1/2} \cdot \mathbf{S}^{\mathcal{AB}} \cdot (\mathbf{S}^{\mathcal{BB}})^{-1/2}) \\ &\quad \mathbf{P}^{\mathcal{A}} = (\mathbf{S}^{\mathcal{AA}})^{-1/2} \cdot \mathbf{U} \quad \text{and} \quad \mathbf{P}^{\mathcal{B}} = (\mathbf{S}^{\mathcal{BB}})^{-1/2} \cdot \mathbf{V}\end{aligned}$$

In practice we use Regularized CCA to make the computation of $\mathbf{P}^{\mathcal{A}}$ and $\mathbf{P}^{\mathcal{B}}$ more robust (see App. A.2). For more details we direct the reader to De Bie et al. (2005) from which this section was inspired.

4. Results

4.1. Experimental Details

We trained VGG11 models (Simonyan & Zisserman, 2015) on CIFAR10 (Krizhevsky & Hinton, 2009), ResNet20 models on CIFAR100 and ResNet18 models on ImageNet (Russakovsky et al., 2015). We trained models of different widths, multiplying their original width by $w \in \{1, 2, 4, 8\}$. The models were trained either using the one-hot encodings of the labels or the CLIP (Radford et al., 2021) embeddings of the class names as training objectives. This last setting is similar to the one used by Stoica et al. (2024) and we found it to yield better learned representations and performances on the task as well as less variability between random initializations.

4.2. CCA's flexibility allows it to better model relations between neurons

We first aim to illustrate the limits of permutation based matching and the flexibility offered by CCA Merge. Suppose we want to merge two models, $\mathcal{A}$ and $\mathcal{B}$, at a specific merging layer, and let $\{\mathbf{z}_i^{\mathcal{M}}\}_{i=1}^n$ denote the set of neurons of model $\mathcal{M} \in \{\mathcal{A}, \mathcal{B}\}$ at that layer. We note here that, in terms of network weights, $\mathbf{z}_i^{\mathcal{M}}$ simply refers to the i-th row of the weight matrix $\mathbf{W}^{\mathcal{M}}$ at that layer. Given the activations of the two sets of neurons in response to a set of given inputs, we can compute the correlation matrix $\mathbf{C}$ where element $\mathbf{C}_{ij}$ is the correlation between neurons $\mathbf{z}_i^{\mathcal{A}}$ and $\mathbf{z}_j^{\mathcal{B}}$. For each neuron $\mathbf{z}_i^{\mathcal{A}}$, for $1 \leq i \leq n$, the distribution of its correlations with all neurons from model $\mathcal{B}$ is of key interest for the problem of model merging. If, as the permutation hypothesis implies, there exists a one-to-one mapping between $\{\mathbf{z}_i^{\mathcal{A}}\}_{i=1}^n$ and $\{\mathbf{z}_i^{\mathcal{B}}\}_{i=1}^n$, then we would expect to have one highly correlated neuron for each $\mathbf{z}_i^{\mathcal{A}}$ – say $z_j^{\mathcal{B}}$ for some $1 \leq j \leq n$ – and all other correlations $\mathbf{C}_{ik}$, $k \neq j$, close to zero. On the other hand, if there are multiple neurons from model $\mathcal{B}$ highly correlated with $z_i^{\mathcal{A}}$, this would indicate that the feature learned by $z_i^{\mathcal{A}}$ is distributed across multiple neurons in model $\mathcal{B}$ – a relationship that CCA Merge would capture.

In the left column of Fig. 2, we plot the distributions of the correlations between two ResNet20x8 models (i.e., all the elements from the correlation matrix $\mathbf{C}$) for 2 different merging layers. The vast majority of correlations have values around zero, as expected, since each layer learns multiple different features. In the right column of Fig. 2 we use box plots to show the values of the top 5 correlation values across all $\{\mathbf{z}_i^{\mathcal{A}}\}_{i=1}^n$. For each neuron $\mathbf{z}_i^{\mathcal{A}}$, we select its top k-th correlation from $\mathbf{C}$ and we plot these values for all neurons $\{\mathbf{z}_i^{\mathcal{A}}\}_{i=1}^n$. For example, for $k = 1$, we take the value $\max_{1 \leq j \leq n} \mathbf{C}_{ij}$, for $k = 2$ we take the second largest value from the i-th row of $\mathbf{C}$, and so on. We observe the top correlations values are all relatively high but none of them approaches full correlation (i.e., value of one), suggesting that the feature learned by each neuron $\mathbf{z}_i^{\mathcal{A}}$ from model $\mathcal{A}$ is distributed across multiple neurons from $\mathcal{B}$ – namely, those having high correlations – as opposed to having a single highly correlated match.

Given the flexibility of CCA Merge, we expect it to better capture these relationships between the neurons of the two networks. We recall that CCA Merge computes a linear transformation $\mathbf{T}$ that matches to each neuron $\mathbf{z}_i^{\mathcal{A}}$ a linear combination $\mathbf{z}_i^{\mathcal{A}} \approx \sum_{j=1}^n \mathbf{T}_{ij} \mathbf{z}_j^{\mathcal{B}}$ of the neurons in $\mathcal{B}$. We expect the distribution of the coefficients (i.e., elements of $\mathbf{T}$) to match the distribution of the correlations ($\mathbf{C}_{ij}$ elements), indicating the linear transformation found by CCA Merge adequately models the correlations and relationships between the neurons of the two models. For each neuron

$\mathbf{z}_i^{\mathcal{A}}$, we select its top k-th, for $k \in \{1, 2\}$, correlation from the i-th row of $\mathbf{C}$ and its top k-th coefficient from the i-th row of $\mathbf{T}$ and we plot a histogram of these values for all neurons $\{\mathbf{z}_i^{\mathcal{A}}\}_{i=1}^n$ in Fig. 3. Indeed, the distributions of the correlations and those of the CCA Merge coefficients are visually similar, albeit not fully coinciding. To quantify this similarity we compute the Wasserstein distance between these distributions, normalized by the equivalent quantity if the transformation was a permutation matrix. For a permutation matrix, the top 1 values would be of 1 for every neuron $\mathbf{z}_i^{\mathcal{A}}$ and all other values would be 0. We can see that CCA Merge finds coefficients that closely match the distribution of the correlations, more so than simple permutations, since the ratio of the two distances are 0.15 and 0.04, respectively, for top 1 values in the two considered layers and 0.35 and 0.23 for top 2 values.

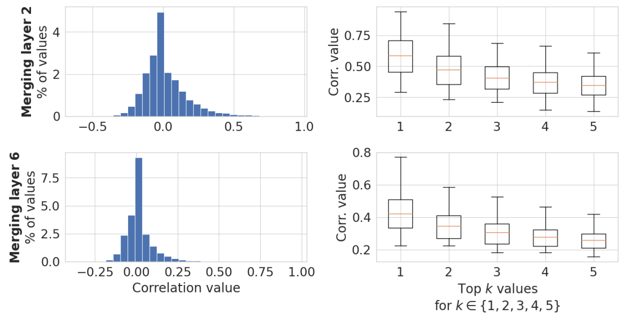

Figure 2: **Left column:** distribution of correlation values between the neurons $\{\mathbf{z}_i^{\mathcal{A}}\}_{i=1}^n$ and $\{\mathbf{z}_i^{\mathcal{B}}\}_{i=1}^n$ of two ResNet20x8 models ($\mathcal{A}$ and $\mathcal{B}$) trained on CIFAR100 at two different merging layers; **Right column:** for $k \in \{1, 2, 3, 4, 5\}$ the distributions of the top k-th correlation values for all neurons in model $\mathcal{A}$ at those merging layers.

Furthermore, when using permutations to merge 2 models, $\mathcal{A}$ and $\mathcal{B}$, a large percentage (25-50%) of the neurons from model $\mathcal{A}$ do not get matched with their highest correlated neuron from model $\mathcal{B}$ by the permutation matrix, we call these non-optimal matches. In such cases, the relationship between these highly correlated but not matched neurons is completely ignored during merging. CCA Merge on the other hand consistently assigns high transformation coefficients to the top correlated neurons. See App. B for more details.

4.3. Models merged with CCA Merge achieve better performance

In Table 1 we show the test accuracies of merged VGG11 and ResNet20 models of different widths trained on CIFAR10 and CIFAR100 respectively for CCA Merge and multiple other popular model merging methods. The number of models being merged is 2 and for each experiment, we report the mean and standard deviation across 4 merges where the base models were trained with different initialization, data augmentation, and data order seeds. Results for more widths can be found in App. C. We report the average accuracies of the base models being merged under the label "Base models avg." (i.e. each model is evaluated individually and their accuracies are then averaged) as well as the accuracies of ensembling the models (the logits of the different models are averaged and the final prediction is the argmax). Ensembling is considered to be the upper limit of what model fusion methods can achieve. Also, since the models being merged were trained on the same data, we do not expect the merged models to outperform the endpoint ones in this particular setting, see App. G for more details. We compare CCA Merge with the following methods:

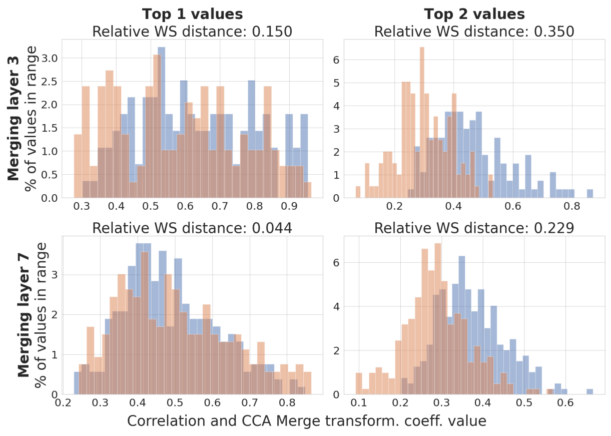

Figure 3: Distributions of top 1 (**left column**) and 2 (**right column**) correlations (blue) and CCA Merge transformation coefficients (orange) across neurons from model $\mathcal{A}$ at two different merging layers. In the left column for example, for each neuron $\mathbf{z}_i^{\mathcal{A}}$ we have one correlation value corresponding to $\max_{1 \leq j \leq n} \mathbf{C}_{ij}$ and one coefficient value corresponding to $\max_{1 \leq j \leq n} \mathbf{T}_{ij}$ where $\mathbf{C}$ is the cross-correlation matrix between neurons of models $\mathcal{A}$ and $\mathcal{B}$, and $\mathbf{T}$ is the CCA Merge transformation matching neurons of $\mathcal{B}$ to those of $\mathcal{A}$. Wasserstein distance between the distributions of top $k \in \{1, 2\}$ correlations and top k Merge CCA coefficients are reported, relative to equivalent distances between correlations and Permute transforms (all top 1 values are 1, and top 2 values are 0).

- **Direct averaging:** averaging the models' weights without applying any transformation to align neurons. This is equivalent to $\mathbf{T} = \mathbf{T}^{-1} = \mathbf{I}$, the identity matrix.
- **Permute:** permuting model weights to align them, the permutation matrix is found by solving the linear sum assignment problem consisting of maximizing the sum of correlations between matched neurons. This method is equivalent to the "Matching Activations" one from Ainsworth et al. (2023), the "Permute" method considered in Stoica et al. (2024) and the neuron alignment algorithm proposed by Li et al. (2015); Tatro et al. (2020) and used in Jordan et al. (2023).

- **OT Fusion:** Using optimal transport to align neurons. This is the method presented in Singh & Jaggi (2020).
- **Matching Weights:** permuting model weights by directly minimizing the distance between the two model weights by solving a sum of bilinear assignments problem (SOBLAP). This is the main method from Ainsworth et al. (2023).
- **ZipIt!:** model merging method proposed by Stoica et al. (2024). We note that ZipIt! also allows the merging of neurons from the same network which results in a redundancy reduction effect that the other methods do not have. Also, it isn't strictly speaking a permutation-based method although similar.

The models were trained from scratch from different initializations, the merging was based on the training data (to compute activation statistics) but all accuracies reported are on the test set. For ResNets, we recompute the BatchNorm statistics after the weight averaging and before evaluation as suggested by Jordan et al. (2023) to avoid variance collapse.

VGG11 models merged with CCA Merge have significantly higher accuracies than models merged with any other method, and this is true across all model widths considered. Differences in accuracy ranging from 10% ($\times 8$ width) up to 25% ($\times 1$ width) can be observed between CCA Merge and the second-best performing method. Furthermore, CCA Merge is more robust when merging smaller width models, incurring smaller accuracy drops than other methods when the width is decreased from $\times 8$ to $\times 1$; 1.71% drop for CCA Merge versus 18.34% for Matching Weights and 8.19% for Permute. Lastly, CCA Merge seems to be more stable across different initializations, the accuracies having smaller standard deviations than all other methods for the same width except for Matching Weights for $\times 8$ width models. We note that for VGG models with width multipliers above $\times 2$, we ran into out-of-memory issues when running ZipIt!, which is why those results are not present. The same conclusions seem to hold for ResNets20 trained on CIFAR100, although the differences in performance here are less pronounced.

In Table 2 we present the performance of merged ResNet18 models of width 4 trained on ImageNet. In this setting we ran into OOM issues with ZipIt! therefore we only compare with Direct averaging, Permute, OT Fusion and Matching Weights, the four of which are significantly outperformed by CCA Merge. CCA Merge reduces the gap between model merging methods and model ensembles.

For both VGG and ResNet architectures as well as for all considered datasets the added flexibility of CCA Merge over permutation-based methods seems to benefit the merged models. Aligning models using linear combinations allows CCA Merge to better model relationships between neurons and to take into account features that are distributed across multiple neurons. In addition to the raw performance benefits, CCA Merge seems to be more stable across different model widths as well as across different random initializations and data order and augmentation.

4.4. CCA merging finds better common representations between many models

In this section, we present our results related to the merging of many models, a significantly harder task. This constitutes the natural progression to the problem of merging pairs of models and is a more realistic setting for distributed or federated learning applications where there are often more than 2 models. Furthermore, aligning populations of neural networks brings us one step closer to finding the common learned features that allow different neural networks to perform equally as well on complex tasks despite having different initializations, data orders, and data augmentations.

As previously mentioned, the problem of merging many models is often ignored by past works except for the settings in which linear mode connectivity is easily obtained. Ainsworth et al. (2023) introduced "Merge Many", an adaptation of Matching Weights for merging a set of models. In Merge Many each model in the set is sequentially matched to the average of all the others until convergence. A simpler way of extending any model merging method to the many models setting is to choose one of the models in the group as the *reference model* and to align every other network in the group to it. Then the reference model and all the other aligned models can be merged. It is by using this "all-to-one" merging that we extend CCA Merge, Permute, OT Fusion and Matching Weights to the many model settings. ZipIt! is naturally able to merge multiple models since it aggregates all neurons and merges them until the desired size is obtained.

In Fig. 4 we show the accuracies of the merged models as the number of models being merged increases. For both VGG and ResNet architectures aligning model weights with CCA continues to yield better performing merged models. In fact, models merged with CCA Merge applied in an all-to-one fashion maintain their accuracy relatively well while the ones merged with other methods see their accuracies drop significantly. In the VGG case, the drop for other methods is drastic, all merged models having less than 20% accuracy when more than 3 models are being merged which constitutes a decrease of more than 25% from the 2 models merged scenario. CCA Merge on the other hand suffers a drop in accuracy of less than 3% when going from merging 2 models to 5, staying around the 80% mark. We also note that despite being designed specifically for the many models setting, Merge Many performs only slightly better than its 2 model counterpart (Matching Weights applied in an all-to-one fashion).

Method	VGG11×1 CIFAR10	VGG11×8 CIFAR10	ResNet20×1 CIFAR100	ResNet20×8 CIFAR100
Base models avg.	87.27 ±0.25%	88.20 ±0.45%	69.21 ±0.22%	78.77 ±0.28%
Ensemble	89.65 ±0.13%	90.21 ±0.24%	73.51 ±0.20%	80.98 ±0.21%
Direct averaging	10.54 ±0.93%	10.45 ±0.74%	1.61 ±0.16%	14.00 ±1.66%
Permute	54.39 ±6.45%	62.58 ±3.31%	28.76 ±2.20%	72.90 ±0.08%
OT Fusion	53.86 ±10.4%	68.32 ±3.13%	29.05 ±2.55%	72.45 ±0.08%
Matching Weights	55.40 ±4.67%	73.74 ±1.77%	21.38 ±4.36%	74.29 ±0.51%
ZipIt!	52.93 ±6.37%	-	25.26 ±2.30%	72.47 ±0.41%
CCA Merge (ours)	**82.65 ±0.73%**	**84.36 ±2.09%**	**31.79 ±1.97%**	**75.06 ±0.18%**

Table 1: VGG11 trained on CIFAR10 & ResNet20 trained on CIFAR100 - Accuracies and standard deviations from 4 different merges of 2 models are presented. Models averaged with CCA Merge notably outperform models merged with other methods, narrowing the gap between merged models and model ensembles. Model ensembles are significantly more memory and compute expensive and represent the upper bound of attainable performance for model merging methods.

Method	Top 1 Acc. (%)	Top 5 Acc. (%)
Base models avg.	75.44 ±0.06	92.17 ±0.05
Ensemble	77.62 ±0.07	93.48 ±0.01
Direct averaging	0.12 ±0.03	0.64 ±0.01
Permute	51.45 ±1.02	76.96 ±0.76
OT Fusion	50.55 ±0.98	76.35 ±0.97
Matching Weights	45.41 ±0.33	72.78 ±0.42
CCA Merge (ours)	**63.61 ±0.22**	**85.41 ±0.25**

Table 2: ResNet18x4 trained on ImageNet - Accuracies and standard deviations from 3 different merges of 2 models are presented. Even on this significantly harder image classification task CCA Merge outperforms past model merging methods for both top 1 and top 5 accuracies, helping to reduce the gap between model merging methods and model ensembles.

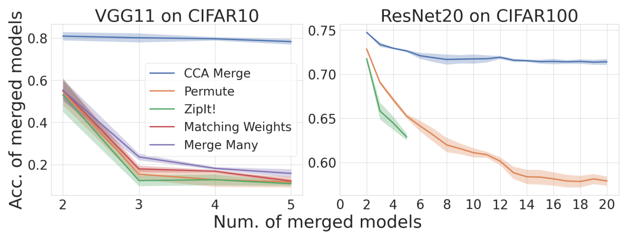

Figure 4: Accuracies of averaging multiple models after feature alignment with different merging methods. Mean and standard deviation across 4 random seeds are shown.

For ResNets, the accuracy of models merged with Permute drops by ~15% when going from merging 2 models to 20. While less drastic than in the VGG case this decrease in performance is still significant. ZipIt! displays a slightly more pronounced drop when going from 2 models merged to 5. CCA Merge on the other hand is a lot more robust, incurring a less than 4% drop in accuracy even as the number of merged models is increased to 20. Additionally, accuracy values for models merged by CCA Merge seem to plateau sooner than those for Permute.

These results suggest that CCA Merge is significantly better than past methods at finding the "common features" learned by groups of neural networks and aligning them. The limitations of permutation-based methods in taking into account complex relationships between neurons from different models are highlighted in this context. Here it is harder to align features given that there are more of them to consider and therefore easier to destroy the features when averaging them.

Permutations and feature mismatches To explain the success of CCA Merge in the multi-model setting, we have examined feature mismatches between networks. When merging models $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, we choose one reference model, say model $\mathcal{A}$, align $\mathcal{B}$ and $\mathcal{C}$ to $\mathcal{A}$ and then merge by averaging. This results in an indirect matching of models $\mathcal{B}$ and $\mathcal{C}$ through the reference model $\mathcal{A}$. Neurons i from $\mathcal{B}$ and j from $\mathcal{C}$ are matched indirectly through $\mathcal{A}$ if they both are matched to the same neuron k from $\mathcal{A}$. Ideally, this indirect matching should be the same as the direct matching resulting from aligning $\mathcal{B}$ and $\mathcal{C}$ directly. This would mean that the same features from $\mathcal{B}$ and $\mathcal{C}$ are matched regardless of whether they are merged together or with a third model. However, this does not seem to be the case for permutation-based methods: over 50% of the neurons from $\mathcal{B}$ don't get matched to the same neuron from $\mathcal{C}$ when aligned directly or indirectly through $\mathcal{A}$. Additionally, the Frobenius norm of the differences between the direct and indirect matching matrices is significantly lower for CCA Merge than for a permutation-based method. This suggests CCA Merge generates fewer feature mismatches in the multi-model setting, explaining in part its success over permutation-based methods. For detailed results and further information see App. E.

Method	(1) 80%-20%	(2) Dirichlet	(3) 50 classes
Base models avg.	65.66 $\pm$0.71%	59.98 $\pm$1.80%	41.42 $\pm$0.54%
Ensemble	77.84 $\pm$0.23%	73.77 $\pm$0.44%	69.91 $\pm$0.49%
Direct averaging	11.40 $\pm$1.62%	20.55 $\pm$3.07%	16.90 $\pm$2.02%
Permute	62.11 $\pm$0.30%	58.45 $\pm$1.76%	43.82 $\pm$1.31%
OT Fusion	61.56 $\pm$0.21%	57.67 $\pm$1.49%	43.02 $\pm$1.27%
Matching Weights	58.18 $\pm$0.68%	55.87 $\pm$1.80%	41.15 $\pm$1.45%
ZipIt!	61.41 $\pm$0.51%	57.97 $\pm$1.29%	**55.08 $\pm$0.70%**
CCA Merge (ours)	**66.35 $\pm$0.19%**	**60.38 $\pm$1.68%**	46.57 $\pm$0.76%

Table 3: ResNet20$\times$8 trained on 3 different splits of CIFAR100 - Accuracies and standard deviations from 4 different merges of 2 models are presented. When the models being merged have learned different features from disjoint sets of the training data but with all the classes (splits 1 and 2) CCA Merge is the only model merging method which outperforms the average of the base models. In the case where the models being merged were trained on disjoint subsets of the classes (split 3) CCA Merge still outperforms past model merging methods except for ZipIt!. However ZipIt! allows same-model neuron merging, making a direct comparison with the other methods, including ours, somewhat unfair.

4.5. CCA Merge is better at combining learned features from different data splits

In this section we consider the more realistic setting where the models are trained on disjoint splits of the data, therefore they're expected to learn (at least some) different features. Such a set-up is natural in the context of federated or distributed learning. We consider ResNet20 models trained on 3 different data splits of the CIFAR100 training dataset. The first (1) data split is the one considered in Ainsworth et al. (2023); Jordan et al. (2023) where one model is trained on 80% of the data from the first 50 classes of the CIFAR100 dataset and 20% of the data from the last 50 classes, the second model being trained on the remaining examples. In the second (2) data split we use samples from a Dirichlet distribution with parameter vector $\alpha = (0.5, 0.5)$ to subsample each class in the dataset and create 2 disjoint data splits, one for each model to be trained on. Lastly, with the third (3) data split we consider the more extreme scenario from Stoica et al. (2024) where one model is trained on 100% of the data from 50 classes, picked at random, and the second one is trained on the remaining classes, with no overlap. For this last setting, in order for both models to have a common output space they were trained using the CLIP (Radford et al., 2021) embeddings of the class names as training objectives. In Table 3 we report mean and standard deviation of accuracies across 4 different model pairs.

For the first two data splits CCA Merge outperforms the other methods, beating the second best method by ~4% and ~2% on the first and second data splits respectively. For the third data split CCA Merge is the second best performing method after ZipIt!. However, ZipIt! was designed for this specific setting and, as we previously noted, it allows the merging of features from the same network to reduce redundancies, thus making it more flexible than the other methods which only perform "alignment". In all cases CCA Merge outperforms or is comparable with the base models average indicating that, to some extent, our method successfully combines different learned features from the two models.

5. Discussion and Conclusion

Recent model fusion successes exploit inter-model relationships between neurons by modelling them as permutations before fusing models. Here, we argue that, while assuming a one-to-one correspondence between neurons yields interesting merging methods, it is rather limited as not all neurons from one network have an exact match with a neuron from another network. Our proposed *CCA Merge* takes the approach of linearly transforming model parameters beyond permutation-based optimization. This added flexibility allows our method to outperform recent competitive baselines when merging pairs of models trained on the same data or on disjoint splits of the data (Tables 1, 2 and 3). Furthermore, when considering the harder task of merging many models, CCA Merge models showcase remarkable accuracy stability as the number of models merged increases, while past methods suffer debilitating accuracy drops. This suggests a path towards achieving *strong linear connectivity* between a set of models, which is hard to do with permutations (Sharma et al., 2024). One limitation of our method is that it requires input data to align the models. The forward pass to compute the activations increases computational costs and in some settings such a "shared" dataset might not be available. We discuss this further in App. F.

Merging many models successfully, without incurring an accuracy drop, is one of the big challenges in this area of research. Our method, CCA Merge, makes a step in the direction of overcoming this challenge. As future work, it would be interesting to further study the common representations learned by populations of neural networks. Also, an interesting future research avenue is to test CCA Merge with models trained on entirely different datasets, to test its limits in terms of combining different features.

Acknowledgements

This work was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) CGS D 569345 - 2022 scholarship [S.H.]; FRQNT-NSERC grant 2023-NOVA-329125 [E.B. & G.W.]; Canada CIFAR AI Chair, NSF DMS grant 2327211 and NSERC Discovery grant 03267 [G.W.]. This work is also supported by resources from Compute Canada and Calcul Québec. The content is solely the responsibility of the authors and does not necessarily represent the views of the funding agencies.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically the sub-field interested in merging models which were trained from scratch, potentially from different initializations, on different data splits or datasets or following different training procedures.

As deep learning has gained in popularity, many open-source models have become available online and are increasingly being used in research and in industry. In many cases these models do not come from the same initialization and the training data and pipelines are varied and in some cases unavailable. Advances in model merging might provide more storage and compute efficient ways of taking advantage of such open-sourced models, thus reducing resource consumption. Furthermore, model merging methods have the potential of allowing us to find the fundamental shared features learned by different models, thus unifying different data representations and furthering our understanding of deep learning models.

Any other societal and ethical impacts of our work are likely to be the same as for any work advancing the field of Machine Learning in general.

References

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

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 233–242. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/arpit17a.html.

De Bie, T., Cristianini, N., and Rosipal, R. *Eigenproblems in Pattern Recognition*, pp. 129–167. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-28247-1. doi: 10.1007/3-540-28247-5_5. URL https://doi.org/10.1007/3-540-28247-5_5.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially no barriers in neural network energy landscape. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1309–1318. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/draxler18a.html.

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

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/frankle20a.html.

Freeman, C. D. and Bruna, J. Topology and geometry of half-rectified network optimization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Bk0FWVcgx.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/be3087e74e9100d4bc4c6268cdbe8456-Paper.pdf.

Goodfellow, I. J. and Vinyals, O. Qualitatively characterizing neural network optimization problems. In Bengio, Y. and LeCun, Y. (eds.), *The third International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6544.

Gotmare, A., Keskar, N. S., Xiong, C., and Socher, R. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r14EOsCqKX.

Ho, T. K. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pp. 278–282. IEEE, 1995.

Horoi, S., Huang, J., Rieck, B., Lajoie, G., Wolf, G., and Krishnaswamy, S. Exploring the geometry and topology of neural network loss landscapes. In Bouadi, T., Fromont, E., and Hüllermeier, E. (eds.), *Advances in Intelligent Data Analysis XX*, pp. 171–184, Cham, 2022. Springer International Publishing. ISBN 978-3-031-01333-1.

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

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. In Globerson, A. and Silva, R. (eds.), *Uncertainty in Artificial Intelligence*, volume 34. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/313.pdf.

Jolicoeur-Martineau, A., Gervais, E., FATRAS, K., Zhang, Y., and Lacoste-Julien, S. Population parameter averaging (PAPA). *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cPDVjsOytS.

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

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 6389–6399. Curran Associates, Inc., 2018.

Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. Convergent learning: Do different neural networks learn the same representations? In Storcheus, D., Rostamizadeh, A., and Kumar, S. (eds.), *Proceedings of the 1st International Workshop on Feature Extraction: Modern Questions and Challenges at NIPS 2015*, volume 44 of *Proceedings of Machine Learning Research*, pp. 196–212, Montreal, Canada, 11 Dec 2015. PMLR. URL https://proceedings.mlr.press/v44/li15convergent.html.

Lobacheva, E., Chirkova, N., Kodryan, M., and Vetrov, D. P. On power laws in deep ensembles. *Advances In Neural Information Processing Systems*, 33:2375–2385, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Morcos, A., Raghu, M., and Bengio, S. Insights on representational similarity in neural networks with canonical correlation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a7a3d70c6d17a73140918996d03c014f-Paper.pdf.

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

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability.

In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Sharma, E., Kwok, D., Denton, T., Roy, D. M., Rolnick, D., and Dziugaite, G. K. Simultaneous linear connectivity of neural networks modulo permutation, 2024. URL https://arxiv.org/abs/2404.06498.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *the 3rd International Conference on Learning Representations (ICLR 2015)*, pp. 1–14, 2015.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22045–22055. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb2697869f56484404c8ceee2985b01d-Paper.pdf.

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

Tatro, N., Chen, P.-Y., Das, P., Melnyk, I., Sattigeri, P., and Lai, R. Optimizing mode connectivity via neuron alignment. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15300–15311. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/aecad42329922dfc97eee948606e1f8e-Paper.pdf.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

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

A. Additional Practical Considerations for CCA Merge

A.1. Aligning $\mathcal{B}$ to $\mathcal{A}$ rather than transforming both

Given that CCA naturally finds a common representation space which maximally correlates the activations of both models being merged, $\mathbf{X}^{\mathcal{A}}$ and $\mathbf{X}^{\mathcal{B}}$, it is natural to consider merging the models in this common space. This would be done by transforming the parameters of model $\mathcal{A}$ with the found transformations $\mathbf{P}_i^{\mathcal{A}}$ and transforming the parameters of model $\mathcal{B}$ with transformations $\mathbf{P}_i^{\mathcal{B}}$ and then averaging both transformed models. However, the reason for applying transformation $\mathbf{T} = \left(\mathbf{P}^{\mathcal{B}} \cdot \mathbf{P}^{\mathcal{A}}\right)^{\top}$ to model $\mathcal{B}$ to align it to $\mathcal{A}$ and merging both models in the representation space of model $\mathcal{A}$ instead of transforming both the weights of model A and model B and merging in the common space found by CCA is because of the ReLU non-linearity. The common representation space found by CCA has no notion of directionality, and might contain important information even in negative orthants (high-dimensional quadrants where at least one of the variables has negative values) that might get squashed to zero by ReLU non-linearities. The representation space of model $\mathcal{A}$ doesn't have this problem.

A.2. Regularized CCA

The closed form CCA solution requires inverting the scatter matrices $\mathbf{S}^{\mathcal{AA}}$ and $\mathbf{S}^{\mathcal{BB}}$ which can lead to poor performance and instability when the eigenvalues of these matrices are small. To make CCA more robust, the identity matrix $\mathbf{I}$ scaled by a regularization hyper-parameter γ is added to the two scatter matrices. Therefore to complete the CCA computation the matrices $\mathbf{S}^{\mathcal{AA}} + \gamma\mathbf{I}$ and $\mathbf{S}^{\mathcal{BB}} + \gamma\mathbf{I}$ are used instead of $\mathbf{S}^{\mathcal{AA}}$ and $\mathbf{S}^{\mathcal{BB}}$.

To chose the hyper-parameter γ for any experiment, i.e. a combination of model architecture and data set / data split, we train additional models from scratch and conduct the merging with different γ values. The γ value leading to the best merged accuracy is kept and applied to the other experiments with CCA Merge. The models used to select γ are discarded to avoid over-fitting and the CCA Merge accuracies are reported for new sets of models.

B. Further empirical analysis of matching matrices

B.1. Non-optimal matches of permutation-based methods

When merging 2 models, $\mathcal{A}$ and $\mathcal{B}$, with Permute, a large percentage (25-50%) of the neurons from model $\mathcal{A}$ do not get matched with their highest correlated neuron from model $\mathcal{B}$ by the permutation matrix, we call these non-optimal matches. In other words, for neurons in $\mathcal{A}$ that have a non-optimal match, a higher correlated neuron from $\mathcal{B}$ exists to which that $\mathcal{A}$ neuron isn't matched. Since permutation-based methods aim to optimize an overall cost, the found solutions match some neurons non-optimally in order to obtain a better overall score. However, since permutation-based methods either completely align two features (value of 1 in the matching matrix) or not at all (value of 0), the merging completely ignores the relationship between these highly correlated but not matched neurons.

In Fig. 5 we present the percent of neurons from net $\mathcal{A}$ that get non-optimally matched to a neuron from net $\mathcal{B}$ for all merging layers inside a ResNet20 trained on CIFAR100. We can see from these results that the limiting nature of permutations causes Permute to disregard some meaningful relationships between learned features, hindering the merged model accuracy.

B.2. Highest correlated neurons are associated to top CCA Merge transformation coefficients

Since CCA Merge combines linear combinations of neurons instead of just individual ones, we can't run the exact same analysis as we did in the previous section for Permute. However, we have looked at whether or not, for each neuron from model $\mathcal{A}$, the top 1 and top 2 correlated neuron from model $\mathcal{B}$ (i.e. neurons with the highest or second highest correlations respectively) get assigned high coefficients in the CCA Merge transformation matrices. We report these values in the Fig. 6.

In the vast majority of cases the highest correlated neuron from model $\mathcal{B}$ gets assigned one of the 5 highest coefficients in the CCA Merge transform, and the top 2 correlated neuron gets assigned one of the 10 highest coefficients of the transform. These results showcase how CCA Merge better accounts for the relationships between neurons of different models during the merging procedure.

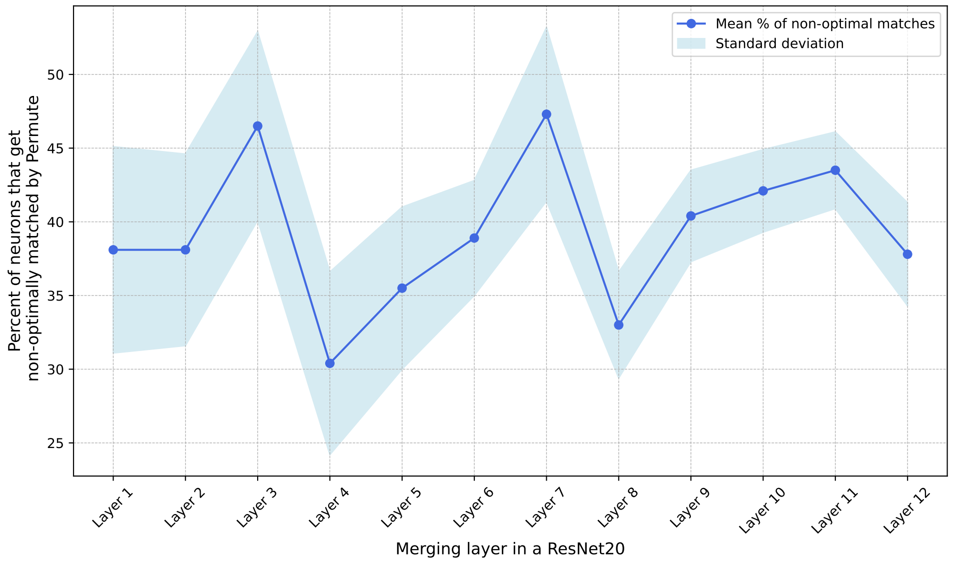

Figure 5: Percent (%) of non-optimal matches when merging ResNet20x8 models trained on CIFAR100. The mean and standard deviation across 15 possible 2-model merges out of a group of 6 models fully trained from different initializations are shown.

C. Additional Results

Here we present the extended Table 1 results, where we also include results for widths $\times 2$ and $\times 4$ for both VGG and ResNet models. The same overall conclusions hold with CCA Merge performing generally better than all other considered baselines. CCA Merge's standard deviation isn't always the lowest however as width increases but it's comparable with the other methods. As noted in the main text we ran into out-of-memory issues when running ZipIt! for VGG models with width greater than $\times 2$.

Table 4: VGG11/CIFAR10 - Accuracies and standard deviations after merging 2 models

Width multiplier	$\times 1$	$\times 2$	$\times 4$	$\times 8$
Base models avg.	87.27 $\pm$0.25%	87.42 $\pm$0.86%	87.84 $\pm$0.21%	88.20 $\pm$0.45%
Ensemble	89.65 $\pm$0.13%	89.74 $\pm$0.44%	90.12 $\pm$0.16%	90.21 $\pm$0.24%
Direct averaging	10.54 $\pm$0.93%	10.28 $\pm$0.48%	10.00 $\pm$0.01%	10.45 $\pm$0.74%
Permute	54.39 $\pm$6.45%	63.32 $\pm$1.12%	64.81 $\pm$1.99%	62.58 $\pm$3.31%
OT Fusion	53.86 $\pm$10.4%	65.97 $\pm$2.13%	66.34 $\pm$3.17%	68.32 $\pm$3.13%
Matching Weights	55.40 $\pm$4.67%	66.98 $\pm$1.96%	71.92 $\pm$2.21%	73.74 $\pm$1.77%
ZipIt!	52.93 $\pm$6.37%	60.73 $\pm$2.07%	-	-
CCA Merge (ours)	**82.65 $\pm$0.73%**	**83.31 $\pm$1.05%**	**85.54 $\pm$0.51%**	**84.36 $\pm$2.09%**

D. VGG Results with REPAIR

In this section, specifically in Table 6, we present the results for merging VGG networks with REPAIR (Jordan et al., 2023) applied to mitigate the variance collapse phenomenon. Since the standard VGG architecture doesn't contain normalization layers it isn't as straightforward as just resetting the BatchNorm statistics as it was for ResNets. Applying REPAIR seems to greatly help past methods and all methods are now comparable in terms of accuracy, with Matching Weights (Ainsworth et al., 2023) being slightly better, but the standard deviations overlapping with CCA Merge. The great performance of CCA Merge without REPAIR suggests that perhaps models merged with our method do not suffer from variance collapse to the same extent as models merged with permutation-based methods. One interesting thing to note is that REPAIR also

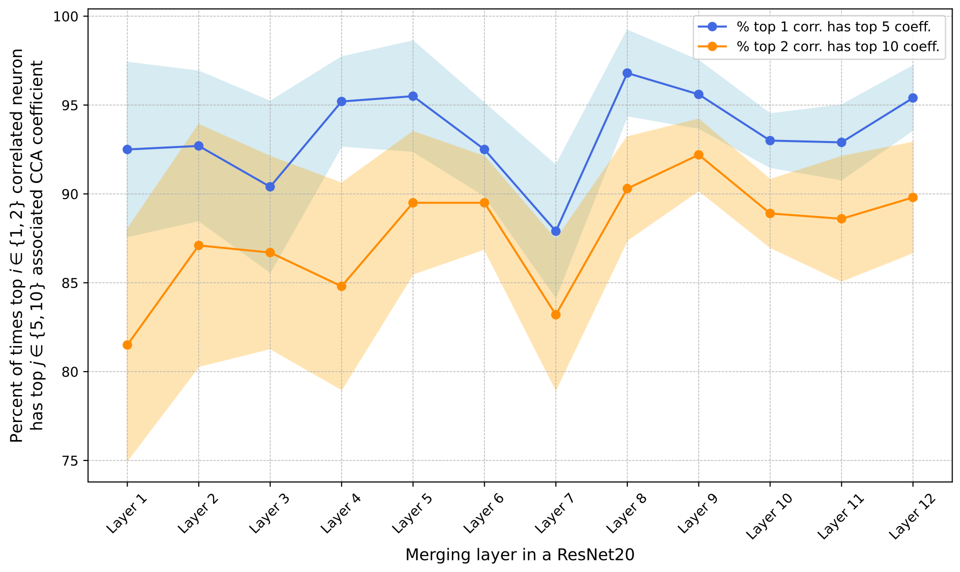

Figure 6: Percent (%) of top 1 and top 2 correlated neurons that have top 5 and top 10 CCA Merge coefficients respectively when merging ResNet20x8 models trained on CIFAR100. The mean and standard deviation across 15 possible 2-model merges out of a group of 6 models fully trained from different initializations are shown.

Table 5: ResNet20/CIFAR100 - Accuracies and standard deviations after merging 2 models

Width multiplier	$\times 1$	$\times 2$	$\times 4$	$\times 8$
Base models avg.	69.21 $\pm$0.22%	74.22 $\pm$0.14%	77.28 $\pm$0.34%	78.77 $\pm$0.28%
Ensemble	73.51 $\pm$0.20%	77.57 $\pm$0.19%	79.90 $\pm$0.08%	80.98 $\pm$0.21%
Direct averaging	1.61 $\pm$0.16%	2.67 $\pm$0.16%	5.13 $\pm$0.52%	14.00 $\pm$1.66%
Permute	28.76 $\pm$2.20%	49.45 $\pm$0.41%	64.65 $\pm$0.34%	72.90 $\pm$0.08%
OT Fusion	29.05 $\pm$2.55%	48.74 $\pm$1.11%	64.07 $\pm$0.38%	72.45 $\pm$0.08%
Matching Weights	21.38 $\pm$4.36%	44.85 $\pm$0.66%	64.66 $\pm$0.45%	74.29 $\pm$0.51%
ZipIt!	25.26 $\pm$2.30%	47.72 $\pm$0.53%	63.69 $\pm$0.31%	72.47 $\pm$0.41%
CCA Merge (ours)	**31.79 $\pm$1.97%**	**54.26 $\pm$1.00%**	**68.75 $\pm$0.22%**	**75.06 $\pm$0.18%**

significantly helps Direct Averaging, where the networks aren't aligned before merging. Without REPAIR, Direct Averaging performed no better than random, however with REPAIR the $\times 8$ models achieve $> 70\%$ accuracy even without any sort of alignment.

Since CCA Merge uses general invertible linear transformations for alignment instead of permutations, the model being aligned can have its functionality altered post alignment. Therefore, when applying REPAIR to models merged with CCA Merge, we do not set the mean and standard deviation of the averaged model's activations to be the average of the mean and standard deviation of the models being merged (standard REPAIR). Instead, we simply reset the mean and standard deviation of the neurons to be the same as the ones of the reference model, i.e. the one to which no transformation is applied and therefore its functionality wasn't altered. To make sure that the comparison to the other methods is fair we have also tried doing this for all the other methods, with the results being around 1% worse than simply applying standard REPAIR (results show in Table 6).

Table 6: VGG11/CIFAR10 - Accuracies and standard deviations after merging 2 models with REPAIR

Width multiplier	$\times 1$	$\times 2$	$\times 4$	$\times 8$
Base models avg.	87.27 ±0.25%	87.42 ±0.86%	87.84 ±0.21%	88.20 ±0.45%
Ensemble	89.65 ±0.13%	89.74 ±0.44%	90.12 ±0.16%	90.21 ±0.24%
Direct averaging	29.89 ±0.58%	43.16 ±5.19%	56.83 ±4.34%	73.75 ±3.12%
Permute	84.56 ±0.30%	85.72 ±0.87%	87.57 ±0.13%	88.35 ±0.62%
OT Fusion	83.33 ±0.32%	84.18 ±1.12%	86.34 ±0.38%	87.09 ±0.30%
Matching Weights	**85.62 ±0.38%**	**86.98 ±0.34%**	**88.68 ±0.18%**	**88.94 ±0.14%**
CCA Merge (ours)	**85.02 ±0.27%**	**86.51 ±0.37%**	87.38 ±0.55%	**88.17 ±0.67%**

E. Why does CCA Merge outperform permutation-based methods in the multi-model setting?

E.1. When merging multiple models, feature mismatches of permutation-based methods get compounded

When merging more than two models, say $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ in the 3-model case, we choose one reference model, WLOG model $\mathcal{A}$, align all other models to that one and then merge by averaging (all-to-one merging). Let $\mathbf{T}_{\mathcal{BA}}$ denote the transformation aligning a given layer of model $\mathcal{B}$ to the same layer of model $\mathcal{A}$ and $\mathbf{T}_{\mathcal{CA}}$ the transformation aligning $\mathcal{C}$ to $\mathcal{A}$ at that same layer, we can also align $\mathcal{A}$ to $\mathcal{B}$ (resp. $\mathcal{C}$) by taking the inverse of $\mathbf{T}_{\mathcal{BA}}$ (resp. $\mathbf{T}_{\mathcal{CA}}$) which we denote $\mathbf{T}_{\mathcal{AB}} = \mathbf{T}_{\mathcal{BA}}^{-1}$ (resp. $\mathbf{T}_{\mathcal{AC}} = \mathbf{T}_{\mathcal{CA}}^{-1}$). This alignment to $\mathcal{A}$ also gives rise to an indirect matching between models $\mathcal{B}$ and $\mathcal{C}$ through $\mathcal{A}$, since we can align $\mathcal{C}$ to $\mathcal{A}$ with $\mathbf{T}_{\mathcal{CA}}$ and then apply the transformation $\mathbf{T}_{\mathcal{AB}}$ to align it to $\mathcal{B}$, we use $\mathbf{T}_{\mathcal{CAB}}$ to denote the transformation of this indirect alignment. In other words, neurons i from $\mathcal{B}$ and j from $\mathcal{C}$ are matched indirectly through $\mathcal{A}$ if they both are matched to the same neuron k from $\mathcal{A}$. To see why multi-model merging fails for permutations we can look at how this indirect matching of $\mathcal{B}$ and $\mathcal{C}$ ($\mathbf{T}_{\mathcal{CAB}}$) compares to the direct matching of $\mathcal{B}$ and $\mathcal{C}$ found by directly optimizing for the transformation matrix $\mathbf{T}_{\mathcal{CB}}$. It turns out that for permutation-based methods these matrices differ significantly, in fact the majority of neurons from net $\mathcal{C}$, between 50-80% on average, do not get matched with the same neuron from $\mathcal{B}$ if we use $\mathbf{T}_{\mathcal{CAB}}$ versus if we use the direct matching $\mathbf{T}_{\mathcal{CB}}$.

In Fig. 7 we present the percent of neurons getting mismatched in this way by Permute for all merging layers inside a ResNet20 trained on CIFAR100. These results help explain why permutations-based methods suffer a drastic drop in accuracy as the number of models being merged increases since there are severe mismatches between the features being aligned that get compounded.

E.2. The direct and indirect matching matrices ($\mathbf{T}_{\mathcal{CB}}$ and $\mathbf{T}_{\mathcal{CAB}}$ resp.) are closer for CCA Merge than for Permute

We can also look directly at the Frobenius norm of the difference between $\mathbf{T}_{\mathcal{CAB}}$ and $\mathbf{T}_{\mathcal{CB}}$ for CCA Merge and Permute to see for which method the indirect matching between $\mathcal{C}$ and $\mathcal{B}$ ($\mathbf{T}_{\mathcal{CAB}}$) and the direct matching between $\mathcal{C}$ and $\mathcal{B}$ ($\mathbf{T}_{\mathcal{CB}}$) are the most similar. We report the results for the same 20 merges considered above in Fig. 8. We see that the indirect and direct matching matrices between $\mathcal{C}$ and $\mathcal{B}$ are significantly closer for CCA Merge than for Permute, which helps explain why CCA Merge outperforms permutation-based methods in the multi-model setting.

We have run these analyses using the Permute matching method since it is the most straightforward to analyze and we have seen from our empirical results that it constitutes a strong baseline. However we expect all permutation-based methods to be susceptible to these types of non-optimal merging and mismatches because of their permutation matching matrices and since their accuracies behave similarly as the number of models being merged increases (for the multi-model scenario).

F. Analysis of computational costs

We have tracked the runtime for 5 different 2-model merges of ResNet20x8 models trained on CIFAR100 and we report these values in the table below.

The merging for methods relying on data (Permute, ZipIt!, CCA Merge) can be split into 2 parts, computing the metrics (eg. covariances or correlations) and computing the transformations. Computing the metrics is by a large margin the most time-expensive part of the procedure, taking on average 33.44s when using the entire CIFAR100 training set. Computing the transforms on the other hand takes less than a second for Permute and CCA Merge and 3.77s for ZipIt!, which represents

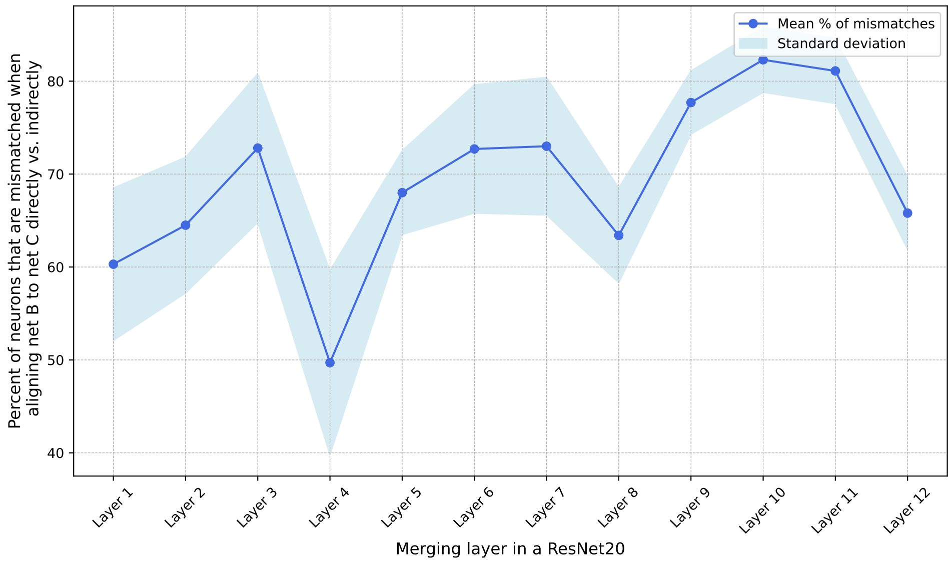

Figure 7: Percent (%) of neurons getting mismatched when merging ResNet20x8 models trained on CIFAR100. The mean and standard deviation across 20 possible 3-model merges out of a group of 6 models fully trained from different initializations are shown.

Table 7: Runtime for 5 different 2-model merges of ResNet20x8 models trained on CIFAR100

	Permute	ZipIt!	CCA Merge	Matching Weights
Time for the whole merging (s)	33.63±0.08	37.39±0.34	34.50±0.57	3.07±0.27
Time for computing the transforms (s)	0.05±0.01	3.77±0.35	0.93±0.57	

$\leq 3\%$ (for Permute and CCA Merge) and 10% (for ZipIt!) of the time required for the entire merging. Among all data-based merging methods, CCA Merge performs the best in accuracy with comparable computation time, therefore it should be prioritized over other such methods. Matching Weights, which doesn't require data, takes 3.07s to complete, however it performs worse than CCA Merge in terms of accuracy in practice.

It is also valuable to describe how these costs scale with model and dataset size. Computing the correlations scales quadratically with the number of neurons in a layer and linearly with the dimension of the activations (which takes into account the size of the input images and the number of examples used to compute the metrics).

G. Merged Models vs. Endpoint Models Accuracies

In past works such as Ainsworth et al. (2023) or Jordan et al. (2023), the merged ResNet models achieving the same accuracy as the endpoint models (or close to) when training on the full train datasets have been extremely wide ones. Specifically, for the ResNet20 architecture on CIFAR10 those results were obtained for models of widths $\times 16$ or greater. Zero barrier merging was not achieved for the model widths considered in our work. Exploring the very wide model setting is not an objective of ours since the effect of model width on merging is already well understood, with wider models leading to better performing merges (Entezari et al., 2022; Ainsworth et al., 2023; Jordan et al., 2023). Therefore we only trained models of width up to $\times 8$, which are more likely to be encountered in practice. For the model widths reported in our work the accuracy barriers are consistent with those reported in Jordan et al. (2023) for Permute (solving the linear sum assignment problem maximizing the correlation between matched neurons), with CCA Merge outperforming those results.

Furthermore, our ResNet20 results are reported on the CIFAR100 dataset which is a harder classification task than CIFAR10, therefore it is harder to achieve zero-barrier merging. Also, in all the disjoint training scenarios presented in Table 3 we do achieve greater accuracy than the endpoint models and outperform past methods, not only in settings considered by past works such as 80%-20% training splits but in novel settings as well, such as Dirichlet training splits.

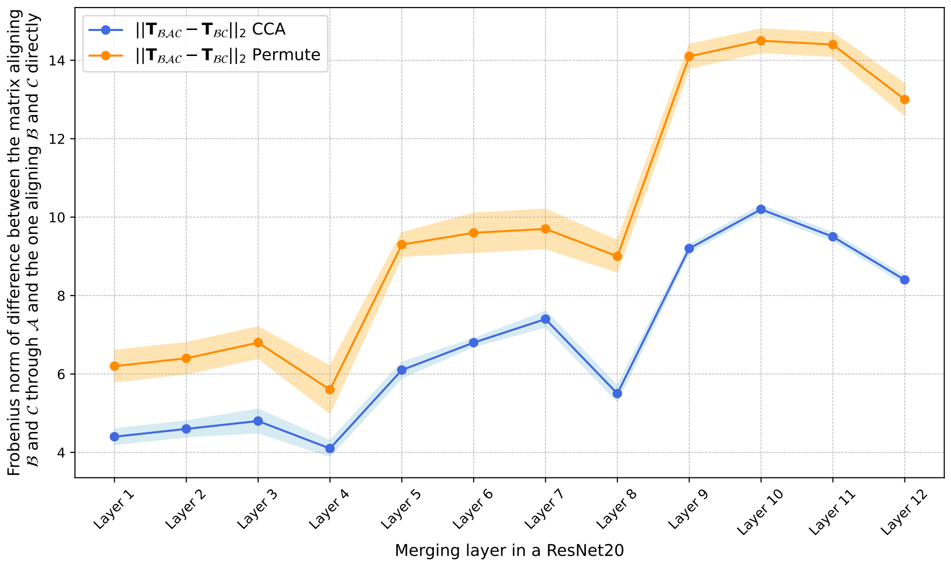

Figure 8: Frobenius norm of the difference between the transformation matrices $\mathbf{T}_{\mathcal{BAC}}$, aligning $\mathcal{B}$ and $\mathcal{C}$ through $\mathcal{A}$, and $\mathbf{T}_{\mathcal{BC}}$ aligning $\mathcal{B}$ and $\mathcal{C}$ directly. The mean and standard deviation across 20 possible 3-model merges out of a group of 6 models fully trained from different initializations are shown.