C^2M^3: Cycle-Consistent Multi-Model Merging

Donato Crisostomi
Sapienza University of Rome
crisostomi@di.uniroma1.it

Marco Fumero
Institute of Science and Technology Austria
fumero@di.uniroma1.it

Daniele Baieri
Sapienza University of Rome
baieri@di.uniroma1.it

Florian Bernard
University of Bonn
fb@uni-bonn.de

Emanuele Rodolà
Sapienza University of Rome
rodola@di.uniroma1.it

Abstract

In this paper, we present a novel data-free method for merging neural networks in weight space. Differently from most existing works, our method optimizes for the permutations of network neurons globally across all layers. This allows us to enforce cycle consistency of the permutations when merging $n \geq 3$ models, allowing circular compositions of permutations to be computed without accumulating error along the path. We qualitatively and quantitatively motivate the need for such a constraint, showing its benefits when merging sets of models in scenarios spanning varying architectures and datasets. We finally show that, when coupled with activation renormalization, our approach yields the best results in the task.

1 Introduction

In the early days of deep learning, modes — parameters corresponding to local minima of the loss landscape — were considered to be isolated. Being depicted as points at the bottom of convex valleys, they were thought to be separated by high-energy barriers that made the transition between them impossible. However, a series of recent works have gradually challenged this perspective, first showing that modes can be actually connected by paths of low energy [10, 14], and later that, in some cases, these paths may even be linear [13]. While linear paths in [13] could only be obtained after training the equally-initialized models for a few epochs, follow-up work [11] speculated that the isolation of modes is a result of the permutation symmetries of the neurons. In fact, given a layer W_ℓ of a fixed network A, a large number of functionally-equivalent networks can be obtained by permuting the neurons of W_ℓ by some permutation P and then anti-permuting the columns of the subsequent layer $W_{\ell+1}$. This intuition led to the conjecture that all modes lie in the same convex region of the parameter space, denoted as *basin*, when taking into account all possible permutations of the neurons of a network.This motivated a series of works trying to align different modes by optimizing for the neuron permutations [1, 21, 29, 36]. This has strong implications for model merging, where different models, possibly trained with different initializations [1, 29, 34] or on different datasets and tasks [1, 36], are aggregated into a single one. In this work, we focus on the *data-free* setting, aligning networks based on some similarity function that is computed directly over the neurons themselves. To this end, we follow Ainsworth et al. [1] and formalize the problem of model merging as an assignment problem, proposing a new algorithm that is competitive with previous approaches while allowing global constraints to be enforced.

The problem We investigate the problem of merging $n > 2$ models, noting that existing pairwise approaches such as [1] do not guarantee cycle consistency of the permutations (see Figure 1). As shown in Figure 2b and Figure 2a, going from a model A to a model C through a model B, and then mapping back to A, results in a different model than the starting one — specifically, the target

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

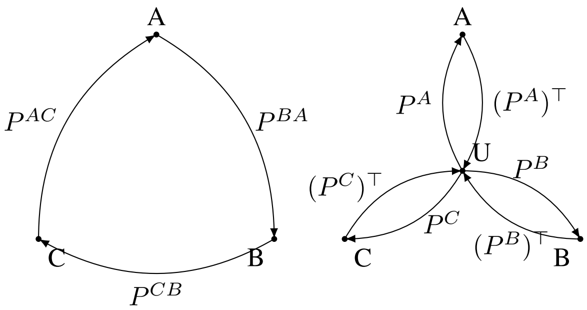


Figure 1: Cycle-Consistent Multi-Model Merging over three models A, B, C. **Left:** existing methods seek pairwise permutations that map between models; note that $P^{AC} \circ P^{CB} \circ P^{BA} \neq I$ in general, unless this is explicitly enforced. **Right:** our method computes permutations P^A, P^B, P^C from each model to a *universe* U, such that a pairwise permutation P^{BA} mapping A to B can be obtained as $P^{BA} = P^B(P^A)^\top$. This way, cycle-consistency is enforced by design and $P^{AC} \circ P^{CB} \circ P^{BA} = I$.

model ends up in a completely different basin. More formally, for these methods, the composition of permutations along any cycle does *not* result in the identity map. This also holds for the $n = 2$ case, where the permutations optimized to align model A to model B are not guaranteed to be the inverse of those mapping B to A; this makes the alignment pipeline brittle, as it depends on an arbitrary choice of a mapping direction.

Contribution To address this issue, we introduce a novel alignment algorithm that works for the general case with $n \geq 2$ models, while *guaranteeing* cycle consistency. The key idea is to factorize each permutation mapping B to A as $P^{AB} = P^A(P^B)^\top$, where $(P^B)^\top$ maps B to a common space denoted as *universe*, and P^A maps from the universe back to A. This formulation ensures cycle consistency by design, as any cyclic composition of such permutations equals the identity.

Our numerical implementation is based on the Frank-Wolfe algorithm [12], and optimizes for the permutations of *all* the layers simultaneously at each step, naturally taking into account the inter-layer dependencies in the process. This desirable property is in contrast with other approaches such as Ainsworth et al. [1], which seek the optimal permutations for each layer separately, and thus can not ensure coherence across the entire network.

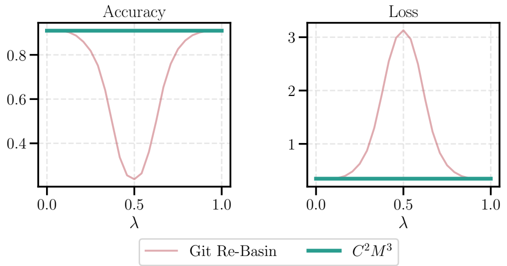
We run an extensive comparison of our approach with existing ones both in the standard pairwise setting and in merging $n > 2$ models, spanning a broad set of architectures and datasets. We then quantitatively measure the influence of architectural width, confirming the existing empirical evidence on its role in linear mode connectivity. Further, we assess how the performance of the merged model depends on the number of models to aggregate, and show that the decay is graceful. We finally analyze the basins defined by the models when mapped onto the universe, and investigate when and to what extent these are linearly connected.

Wrapping up, our contributions are four-fold:

- We propose a new data-free weight matching algorithm based on the Frank-Wolfe algorithm [12] that globally optimizes for the permutations of all the layers simultaneously;
- We generalize it to the case of $n \geq 2$ models, enforcing guaranteed cycle-consistency of the permutations by employing a universal space as a bridge;
- We leverage the multi-model matching procedure for model merging, using the universal space as aggregation point;
- We conduct an extensive analysis showing how the merge is affected by the number of models, their width and architecture, as well as quantitatively measuring the linear mode connectivity in the universe basin.

Finally, to foster reproducible research in the field, we release a modular and reusable codebase containing implementations of our approach and the considered baselines.[1]

[1] https://github.com/crisostomi/cycle-consistent-model-merging



Permutation	Git Re-Basin	C^2M^3
$d\left(A, P_{A\to B\to C\to A}(A)\right)$	41.07	0.0
$d\left(B, P_{B\to C\to A\to B}(B)\right)$	41.18	0.0
$d\left(C, P_{C\to A\to B\to C}(C)\right)$	41.19	0.0

(a) Loss and accuracy curves for a model A and the model mapped back after a cyclic permutation. Models cyclically permuted with `Git Re-Basin` end up in a different basin than the one they started from.

(b) Accumulated error obtained when cyclically permuting models A, B and C as in Figure 1. $P_{A\to B\to C\to A}$ refers to the composition $P_{AC} \circ P_{CB} \circ P_{BA}$ and $d(\cdot)$ is the ℓ_2 loss.

Figure 2: Existing methods accumulate error when cyclically mapping a model through a series of permutations, while C^2M^3 correctly maps the model back to the starting point.

2 Background

Mode connectivity As introduced in Section 1, mode connectivity studies the geometry of the loss landscape with a particular interest on the regions corresponding to local minima. Following Frankle et al. [13], we assess the connectivity for two given modes by computing their loss barrier:

Definition 2.1. (*Loss barrier*) Given two points Θ_A, Θ_B and a loss function $\mathcal{L}$ such that $\mathcal{L}\left(\Theta_A\right) \approx \mathcal{L}\left(\Theta_B\right)$, the *loss barrier* is defined as

$$\max_{\lambda\in[0,1]} \mathcal{L}\left((1-\lambda)\Theta_A + \lambda\Theta_B\right) - \frac{1}{2}\left(\mathcal{L}\left(\Theta_A\right) + \mathcal{L}\left(\Theta_B\right)\right).$$

Intuitively, this quantity measures the extent of the loss increase when linearly moving from the basin of a mode to the other. When two modes share the same basin, the loss does not increase at all and results in a barrier close to zero.

Weight-space symmetries Following the rich line of works on mode connectivity and model merging [1, 11, 13, 29, 36], we start from the essential insight of *neuron permutation invariance* in neural networks. Let us focus on the simple case of a Multi-Layer Perceptrons (MLP), where we can write the computation for an intermediate layer $W_\ell \in \mathbb{R}^{d_{\ell+1}\times d_\ell}$ as $z_{\ell+1} = \sigma\left(W_\ell z_\ell + b_\ell\right)$, with z_ℓ being the input at the ℓ-layer and σ denoting an element-wise activation function. For the sake of a clear exposure, we consider the bias $b_\ell = 0$ in the following. If apply a permutation matrix $P \in \mathbb{P}$ to the rows of the W_ℓ matrix (*i.e.* the neurons), we obtain $z'_{\ell+1} = \sigma\left(PW_\ell z_\ell\right)$. Being an element-wise operator, σ commutes with P and can be neglected wlog. Since $z'_{\ell+1} \neq z_\ell$ when $P \neq I$, we can still nullify the effect of the permutation by anti-permuting the columns of the subsequent layer for the inverse permutation of P, *i.e.* $P^\top$. In fact,

$$z'_{\ell+2} = W_{\ell+1}P^\top z'_{\ell+1} = W_{\ell+1}\underbrace{P^\top P}_{I}W_\ell z_\ell = z_{\ell+2}$$

making pairs of models that only differ by a permutation of the neurons de facto functionally equivalent. Given the enormous number of such permutations, it stands to reason that the resulting weight-space symmetries act as a major factor in the isolation of modes.

Solving for the permutation Given the above considerations, Entezari et al. [11] speculated that all models end up in a single basin after having accounted for permutation symmetries. Assuming this to hold at least in practical cases, Ainsworth et al. [1] proposed a simple algorithm to find the permutations matching two models by maximizing a local version of the sum of bi-linear problems:

$$\arg\max_{\{P_\ell\in\mathbb{P}\}} \sum_{\ell=1}^{L} \langle W_\ell^A, P_\ell W_\ell^B P_{\ell-1}^T \rangle\,, \tag{1}$$

with $P_0 := I$. Noting that Equation (1) is NP-hard, Ainsworth et al. [1] tackle this problem by considering one layer at a time, relaxing the bi-linear problems to a set of linear ones that can be efficiently solved with any Linear Assignment Problem (LAP) solver, e.g., the Hungarian algorithm. This layer-wise linearization of the objective function, however, corresponds to high variance in the results that depend on the random order of the layers during optimization. See Table 7 for an empirical evaluation confirming this issue.

Renormalizing the activations Notwithstanding the quality of the obtained matching, the loss barrier can still be high due to the mismatch in the statistics of the activations. In fact, `REPAIR` [21] empirically shows the presence of a decay in the variance of the activations after the interpolation. They further show that the loss can be drastically reduced by "repairing" the mean and variance of the activations, forcing the statistics of the merged network to interpolate those of the endpoint networks. We refer the reader to Appendix A.4 for an in-depth explanation.

3 Approach

We now propose a novel algorithm to tackle the weight matching problem, first introducing its formulation in the pairwise case and then generalizing it to match and merge a larger number n of models in a cycle-consistent fashion.

Pairwise matching As we have seen, the NP-hardness of Equation (1) demands for a relaxation of the problem to be tackled. Differently from Ainsworth et al. [1], we opt to maintain the objective global with respect to the layers and instead iteratively optimize its linear approximation via the the Frank-Wolfe algorithm [12]. This procedure requires the computation of the gradient of Equation (1) with respect to each permutation P_i, thus we have to account for two contributions for each ∇_{P_i}, *i.e.*, its gradient from permuting the rows of W_i and the one from permuting the columns of W_{i+1}:

$$\nabla_{P_i} f = \underbrace{W_i^A P_{i-1} (W_i^B)^\top}_{\text{from permuting rows}} + \underbrace{(W_{i+1}^A)^\top P_{i+1} W_{i+1}^B}_{\text{from permuting columns}}. \tag{2}$$

The Frank-Wolfe algorithm then uses the gradient to iteratively update the solution by linearly interpolating between the current solution and the projected gradient. We refer to Lacoste-Julien [24] for theoretical guarantees of convergence. The full algorithm is reported in Appendix A.2.

Generalization to n models In order to generalize to n models, we jointly consider all pairwise problems

$$\arg\max_{P_i^{pq}\in\mathbb{P}} \sum_{p=1}^{n} \sum_{\substack{q=1\\ q\neq p}}^{n} \sum_{i=1}^{L} \langle W_i^p, P_i^{pq} W_i^q (P_{i-1}^{pq})^\top \rangle, \tag{3}$$

where the superscript pq indicates that the permutations maps model q to model p, with $P_0^{pq} := I$. In order to *ensure cycle consistency by construction* we replace the quadratic polynomial by a fourth-order polynomial. Dropping the layer subscript for the sake of clear exposure, we replace the pairwise matchings P^{pq} in the objective of Equation (3) by factorizing the permutations into *object-to-universe matchings* $P^{pq} = P^p \circ (P^q)^\top$ so that each model q can be mapped back and forth to a common universe u with a permutation and its transpose, allowing to map model q to model p by composition of $(P^q)^\top$ ($q \to u$) and P^p ($u \to p$). This way, the objective of Equation (3) becomes

$$\sum_{p\neq q}^{n} \sum_{i=1}^{L} \langle W_i^p, P_i^p (P_i^q)^\top W_i^q (P_{i-1}^p (P_{i-1}^q)^\top)^\top \rangle = \sum_{p\neq q}^{n} \sum_{i=1}^{L} \langle (P_i^p)^\top W_i^p P_{i-1}^p, (P_i^q)^\top W_i^q P_{i-1}^q \rangle. \tag{4}$$

As stated by Theorem 3.1, the permutations we obtain using Equation (4) are cycle consistent. We refer the reader to Bernard et al. [5] for the proof and a complete discussion of the subject.

Theorem 3.1. *Given a set of n models $p_0, \dots, p_n$ and object-to-universe permutations $P_i^{p_j}$ computed via Equation (4), the pairwise correspondences defined by $P_i^{p_l p_j} = P_i^{p_l} \circ (P_i^{p_j})^T$ are cycle-consistent, i.e.,*

$$P_i^{p_1 p_j} \circ \cdots \circ P_i^{p_3 p_2} \circ P_i^{p_2 p_1} = I$$

for all layer indices i, $2 \leq j \leq n$.

Similarly to the pairwise case, the approach requires computing the gradients for the linearization. This time, however, each $\nabla_{P_i^A} f$ has four different contributions: one from permuting the rows of its corresponding layer, one from anti-permuting the columns of the subsequent layer, and two other contributions that arise from the symmetric case where A becomes B. In detail,

$$\nabla_{P_\ell^A} = \nabla_{P_\ell^A}^{\text{rows}} + \nabla_{P_\ell^A}^{\text{cols}} + \nabla_{P_\ell^A}^{\text{rows},\leftrightarrows} + \nabla_{P_\ell^A}^{\text{cols},\leftrightarrows} \tag{5}$$

where

$$\nabla_{P_\ell^A}^{\text{rows}} = W_\ell^A P_{\ell-1}^A (P_{\ell-1}^B)^\top (W_\ell^B)^\top P_\ell^B \qquad \nabla_{P_\ell^A}^{\text{cols}} = (W_{\ell+1}^A)^\top P_{\ell+1}^A \, (P_{\ell+1}^B)^\top \, W_{\ell+1}^B \, P_\ell^B$$
$$\nabla_{P_\ell^A}^{\text{rows},\leftrightarrows} = W_\ell^B P_{\ell-1}^B (P_{\ell-1}^A)^\top (W_\ell^A)^\top P_\ell^A \qquad \nabla_{P_\ell^A}^{\text{cols},\leftrightarrows} = (W_{\ell+1}^B)^\top P_{\ell+1}^B \, (P_{\ell+1}^A)^\top \, W_{\ell+1}^A \, P_\ell^A$$

See Algorithm 1 for a complete description of the procedure.

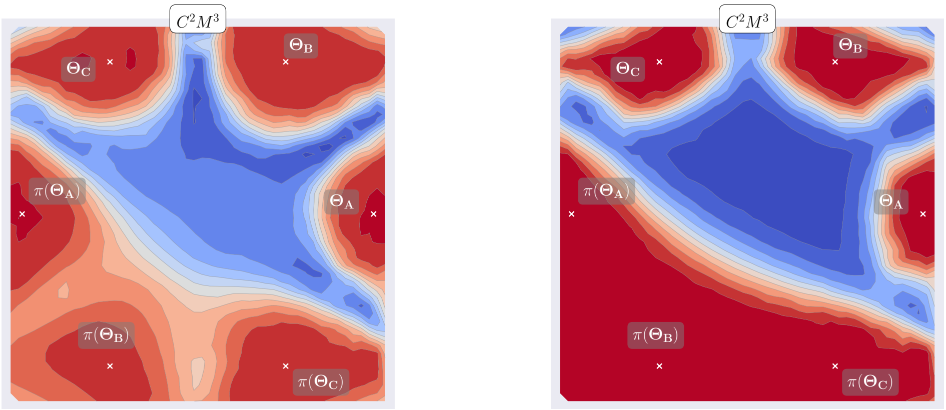
Algorithm 1 Frank-Wolfe for n-Model Matching

```
Require: Weights of $n$ models $M_{i=1}^N$
    tolerance $\epsilon > 0$
Ensure: Approximate solution to Equation (4)
$\mathbf{P}^k \leftarrow$ identity matrices
repeat
    for $(p, q) \in [1, \dots, n] \times [1, \dots, n]$ do
        for $i = 1$ to $L$ do
            $P_i^{p,k}, P_{i-1}^{p,k} \leftarrow$ permutations over rows and columns of $W_i^p$ respectively
            $P_i^{q,k}, P_{i-1}^{q,k} \leftarrow$ permutation over rows and columns of $W_i^q$ respectively
            $\nabla_{P_i^{p,k}} f \pm (W_{\ell+1}^p)^\top P_{\ell+1}^p \, (P_{\ell+1}^q)^\top \, W_{\ell+1}^q \, P_\ell^q$
            $\nabla_{P_{i-1}^{p,k}} f \pm (W_{\ell+1}^p)^\top P_{\ell+1}^p \, (P_{\ell+1}^q)^\top \, W_{\ell+1}^q \, P_\ell^q$
        end for
    end for
    for $P_i^k \in \mathbf{P}^k$ do
        $\Pi_i \leftarrow \text{LAP}(\nabla_{P_i^K} f)$
    end for
    $\alpha \leftarrow$ line search$(f, \mathbf{P}^k, \mathbf{\Pi})$
    for $P_i^k \in \mathbf{P}^k$ do
        $P_i^{k+1} = (1 - \alpha) P_i^k + \alpha \, \Pi_i$
    end for
until $\|f(A, B, \mathbf{P}^{k+1}) - f(A, B, \mathbf{P}^k)\| < \epsilon$
return $\mathbf{P}^k$
```

Merging in the universe space Looking at the loss landscape resulting from interpolating models in Figure 3, we see that the loss curves are much lower when the models are interpolated in the universe space. In fact, the originally disconnected modes end up in the same basin when mapped onto the universe, making it suitable to average the models. Therefore, our merging method aggregates the models by taking the mean of the weights in the universe space, as detailed in Algorithm 2.

Algorithm 2 C^2M^3: Cycle-Consistent Multi Model Merging

```
Require: $N$ models $A_1, \dots, A_N$ with $L$ layers
Ensure: merged model $M$
$\{P_1, \dots, P_N\} \leftarrow$ Frank-Wolfe$(M_1, \dots, M_N)$
for $i = 1$ to $N$ do
    $M_i^{\text{uni}} \leftarrow$ map_to_universe$(A_i, P_i)$
end for
$M^{\text{uni}} \leftarrow \frac{1}{N} \sum_{i=1}^N M_i^{\text{uni}}$
return $M^{\text{uni}}$
```



(a) `ResNet20` over `CIFAR100`.

(b) `MLP` over `MNIST`.

Figure 3: 2D projection of the loss landscape when matching three modes $\Theta_A, \Theta_B, \Theta_C$; the models $\pi(\Theta_A), \pi(\Theta_B), \pi(\Theta_C)$ are their resulting images in the universe, and lie in the same basin. Red zones indicate low-loss regions (typically basins), while blue zones indicate high-loss ones.

4 Experiments

We now evaluate the quality of our proposed framework both in matching models and in the subsequent merging operation. Approaches suffixed with a † indicate the application of `REPAIR`.

Matching and merging two models As described in Section 3, our formalization can readily be used to match $n = 2$ models. In this case, the energy is given by Equation (1) and the permutations are not factorized. We compare the performance of our approach against the `Git Re-Basin` algorithm [1] and the `naive` baseline that aggregates the models by taking an unweighted mean on the original model weights without applying any permutation. In this setting, our method performs on par with the state-of-the-art. Differently from the latter, however, we do not depend on the random choice of layers, as the optimization is performed over all layers simultaneously. As presented in Figure 4, this results in `Git Re-Basin` exhibiting variations of up to 10% in accuracy depending on the optimization seed, while our method shows zero variance. We refer the reader to Appendix B.1 for a thorough evaluation of C^2M^3 over a set of different datasets and architectures. In summary, *our approach is able to match two models with the same accuracy as the state-of-the-art, while being deterministic and independent of the random choice of layers*.

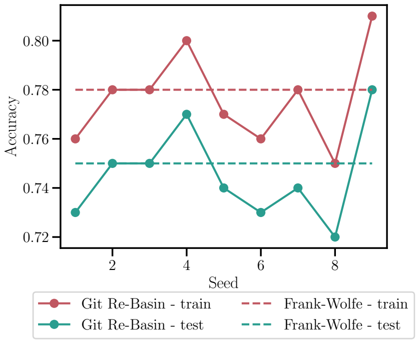


Figure 4: Accuracy of the interpolated model using `Git Re-Basin` [1] over different optimization seeds.

Matching and merging n models We now evaluate C^2M^3 in matching and merging n models. The matching is given by the factorized permutations obtained by Algorithm 1. We compare against two baselines: the simple approach of naively averaging the weights without any matching, and the `MergeMany` approach proposed by Ainsworth et al. [1]. The latter is reported in Appendix A for convenience. As reported in Table 1, C^2M^3 obtains far superior results in terms of accuracy and loss in all considered settings, with accuracy gains as high as $+20\%$. Moreover, our approach natively yields cycle-consistent permutations: Figure 2b shows that `Git Re-Basin` [1] accumulates significant error when computing the distance between the source model and the model obtained by applying a cyclic series of permutations, while our approach is able to perfectly recover the source model. This is further confirmed in Figure 2a, where we show the loss and accuracy curves when interpolating between a model A and the model mapped back after a cyclic permutation.

Matcher		EMNIST					CIFAR10					CIFAR100			
		Accuracy (↑)		**Loss** (↓)			**Accuracy** (↑)		**Loss** (↓)			**Accuracy** (↑)		**Loss** (↓)	
		train	test	train	test		train	test	train	test		train	test	train	test
Naive	MLP	0.03	0.03	3.28	3.28	ResNet 2×	0.10	0.10	3.07	3.07	ResNet 4×	0.01	0.01	5.30	5.30
MergeMany		0.88	0.86	1.11	1.13		0.38	0.37	2.08	2.06		0.31	0.28	3.01	2.76
MergeMany†		0.88	0.86	1.11	1.13		0.50	0.50	2.34	2.30		0.24	0.22	3.31	3.12
C^2M^3		0.89	0.87	1.07	1.10		0.42	0.40	2.11	2.05		0.34	0.30	2.94	2.63
$C^2M^{3\dagger}$		**0.89**	**0.87**	**1.07**	**1.10**		**0.72**	**0.69**	**1.26**	**1.12**		**0.53**	**0.46**	**2.13**	**1.67**
Naive	ResNet 2×	0.04	0.04	4.04	4.04	VGG16	0.10	0.10	2.31	2.31	ResNet 16×	0.01	0.01	6.22	6.22
MergeMany		0.03	0.03	7.17	7.18		0.10	0.10	2.36	2.36		0.45	0.38	2.32	3.06
MergeMany†		0.03	0.03	4.74	4.72		0.60	0.57	1.43	1.32		0.41	0.35	2.27	2.68
C^2M^3		0.27	0.27	3.43	3.47		0.11	0.11	2.34	2.34		0.46	0.39	2.25	3.03
$C^2M^{3\dagger}$		**0.60**	**0.60**	**1.32**	**1.34**		**0.64**	**0.62**	**1.34**	**1.23**		**0.60**	**0.49**	**1.43**	**2.23**

Table 1: Accuracy of the merged model when merging 5 models trained with different initializations. The best results are highlighted in bold. † denotes models after the REPAIR operation.

Models cyclically permuted with Git Re-Basin end up in a different basin than the one they started from, while our cycle-consistent approach ensures that the target model is exactly the same as the source. Wrapping up, *our approach matches and merges n models with a significant improvement in performance over the state-of-the-art, while ensuring cycle-consistent permutations.*

Model similarity before and after mapping As we can see in Figure 5, the cosine similarity of the weights of the models is 3× higher after mapping the latter to the universe. This suggests that the initial quasi-orthogonality of models is at least partially due to neuron permutation symmetries. We also report in Appendix C.1.2 the similarity of the representations between pairs of models. Interestingly, the latter does not change before and after mapping to the universe, but only if we consider a similarity measure that is invariant to orthogonal transformations such as CKA [22]. When using a measure that does not enjoy this property, such as the Euclidean distance, the representations become much more similar in the universe space. In short, *the models are 3× more similar in the universe space and the mapping affects the representations as an orthogonal transformation.*

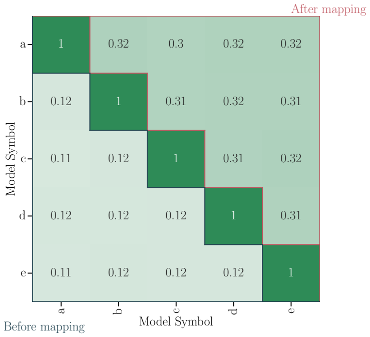


Figure 5: Cosine similarity of the weights of 5 ResNet20 trained on CIFAR10 with 2× width.

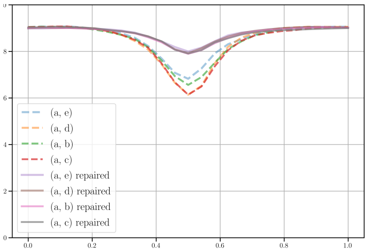
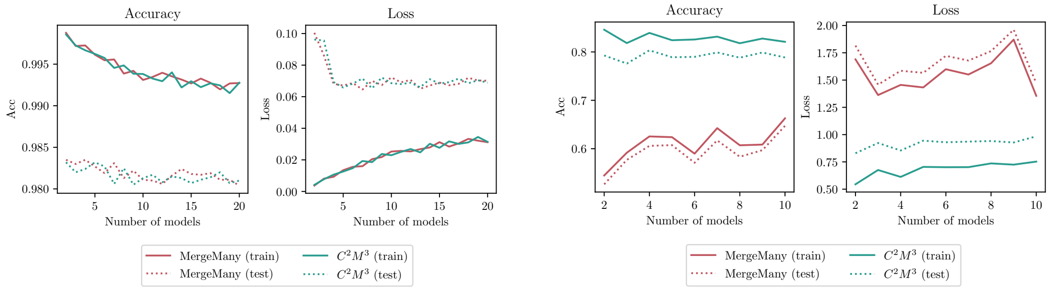


Figure 6: Interpolation curves of VGG models in the universe.

Effect of activation renormalization Our empirical evidence also points out the benefits of the REPAIR operation [21] that is performed after the merging. In fact, the detrimental effect of model averaging on the activation statistics [21] still applies when taking the mean of n models instead of two. Our results clearly show the benefit of REPAIR, making it a key ingredient of our overall framework. Requiring meaningful interpolation endpoints to be effective, REPAIR has lower benefit when employed on the MergeMany algorithm of Ainsworth et al. [1]. In fact, iteratively taking means of different random model subsets and aligning the left-out models to the mean is a more complex process than interpolating between some endpoint models. By taking the mean of models in the universe space, we are instead effectively interpolating between endpoint models that can be used for the computation of the statistics in Equation (8). Figure 6 shows the benefit of using the repair operation on 5 VGG models trained on CIFAR10 mapped to the universe space. Specifically we fix one model "a" and we linearly interpolate in the universe space with respect to the other models, measuring accuracy before and after applying REPAIR. Other than boosting performance, we observe that the latter reduces the variance over interpolation paths, resulting in the interpolation curves of all the models overlapping. Overall, *using the models in the universe as meaningful endpoints to gather activation statistics, our approach can fully leverage activation renormalization techniques such as REPAIR.*



(a) MLPs trained over `MNIST`.

(b) `ResNet20` models trained over `CIFAR10`.

Figure 7: Accuracy and loss when increasing the number n of models to match and merge.

Increasing n In this experiment, we show how the merged model behaves when increasing the number of aggregated models. As we can see in Figure 7a, increasing the number of MLPS up to 20 causes the performance to slightly deteriorate in a relative sense, but remaining stable in an absolute sense as it doesn't fall below 98%. More surprisingly, Figure 7b shows that for a `ResNet20` architecture with $4\times$ width the loss and accuracy are not monotonic, but rather they seem to slightly fluctuate. This may hint at the merging process being more influenced by the composition of the model set, than by its cardinality. Intuitively, a model that is difficult to match with the others will induce a harder optimization problem, possibly resulting in a worse merged model. We dive deeper in the effect of the composition of the set of models in Appendix C.2. In short, *our approach is effective in merging a larger number of models, suggesting promise in federated settings.*

Varying widths We now measure how architectural width affects model merging, taking into consideration `ResNet20` architectures with width $W \in \{1, 2, 4, 8, 16\}$. As we can see in Figure 8, *width greatly increases the performance of the merged model*, reaching the zero-loss barrier first observed in [1] when $W = 16$. This is in line with the observations relating linear mode connectivity and network widths [11, 1], and confirms the intuition that the merging is only effective when modes *can* be linearly connected.

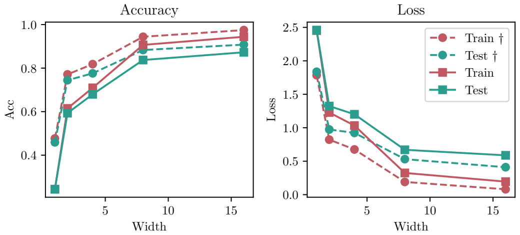


Figure 8: Accuracy and loss when merging 3 `ResNet20`s trained over `CIFAR10` with different widths. † indicates models after applying `REPAIR`.

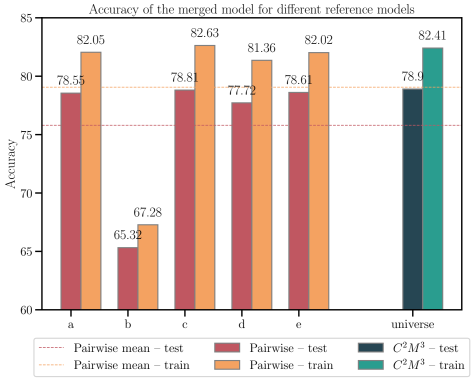
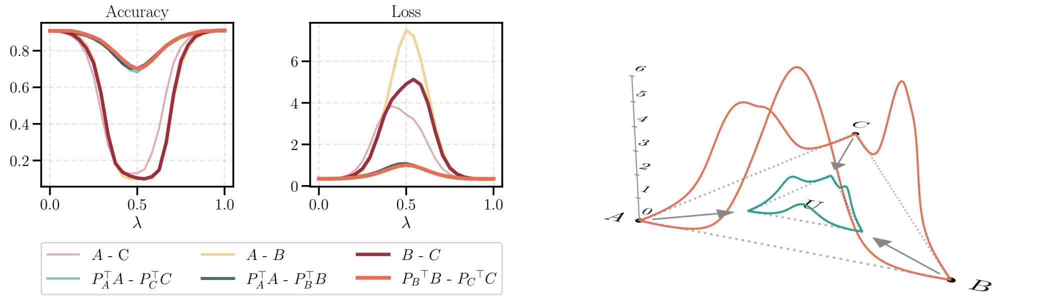


Figure 9: Accuracy of the merged model when mapping towards one arbitrary model (a, b, c, d, e) versus using C^2M^3 and the universe space.

Alternative: fixing one model as universe Alternatively, one could achieve cycle consistency by using one of the source models as reference and learning pairwise maps towards this one. This, however, would require arbitrarily selecting one of the models, making the overall merging dependent on an arbitrary choice. To see why this matters, we merged 5 `ResNet20-4`$\times$ by choosing one model as reference and aggregating the models in its basin. Figure 9 shows severe oscillations in the results, with one model reaching an accuracy as low as 65%, while our approach performs as the best possible reference. This approach, moreover, does not address multi-model merging, as it is intrinsically pairwise: in a multi-task setting, models optimally mapped to a reference basin would only be able to solve the task solved by the reference model. This would prevent merging to be used for models containing complementary information, such as knowledge fusion [19]



(a) 2D visualization of accuracy and loss of the models sampled from the pairwise interpolation lines.

(b) 3D visualization of the loss of the models sampled from the pairwise interpolation lines.

Figure 10: Linear mode connectivity before and after mapping to the universe for 3 `ResNet20-2`× models trained over `CIFAR10` according to Algorithm 1.

or multi-task merging [36]. In our setting, instead, the universe model must by design be a function of all the models and act as a midpoint, hence aggregating information from all the models.

Linear mode connectivity in the universe Figure 10 shows that the loss curves of models interpolated in the universe are much lower than those interpolated in the original space, suggesting that the models are more connected in the former. These results, together with the loss landscape observed in Figure 3, *encourage merging the models in the universe space due to the lower loss barrier.*

5 Related work

Mode Connectivity and model merging. Mode connectivity studies the weights defining local minima. Frankle et al. [13] studied linear mode connectivity of models that were trained for a few epochs from the same initialization and related it to the lottery ticket hypothesis. Without requiring the same initialization, Entezari et al. [11] speculated that all models share a single basin after having solved for the neuron permutations. Model merging aims at aggregating different models into a single one to inherit their capacities without incurring in the cost and burden of ensembling. In this regard, Singh and Jaggi [34] proposed an optimal-transport based weight-matching procedure, while `Git Re-Basin` [1] proposed three matching methods and the `MergeMany` procedure seen in Section 4. Subsequently, `REPAIR` [21] showed that a significant improvement in performance of the interpolated model may be obtained by renormalizing its activactions rather than changing matching algorithm. Differently from all these works, we consider merging n models and propose a principled way to perform it with cycle-consistency guarantees.

Cycle consistency. Ensuring cycle consistency of pairwise maps is a recurring idea in the computer vision and pattern recognition literature. In the realm of deep learning, earlier studies addressing multi-graph matching achieved cycle consistency by synchronizing ex-post the predicted pairwise permutations [40, 41]. The alternative approach using an object-to-universe matching framework, which we adopt here, inherently ensures cycle consistency by construction, as demonstrated in [4, 16, 31]. To the best of our knowledge, none of the existing works tackles cycle-consistent alignment of neural models. We refer to Appendix A.1 for a more detailed list of related works.

6 Conclusions

In this work, we treated the problem of model matching and merging. We first introduced a novel weight matching procedure based on the Frank-Wolfe algorithm that optimizes for the permutation matrices of all layers jointly, and then generalized it to the case of n models. Guaranteeing cycle-consistency, the latter poses a principled way to merge a set of models without requiring an arbitrary reference point. We then showed the approach to yield superior performance compared to existing ones in merging multiple models in a set of scenarios spanning different architectures and datasets. We believe the formalism to elegantly fit the requirement for the merging operation to unify the different models into a cohesive one, rather than mapping all of them to one of the models in the set.

Acknowledgments

This work is supported by the ERC grant no.802554 (SPECGEO), PRIN 2020 project no.2020TA3K9N (LEGO.AI), and PNRR MUR project PE0000013-FAIR. Marco Fumero is supported by the MSCA IST-Bridge fellowship which has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 101034413. We thank Simone Scardapane for the helpful feedback on the paper.

References

[1] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2022.

[2] Federica Arrigoni and Andrea Fusiello. Synchronization problems in computer vision with closed-form solutions. *International Journal of Computer Vision*, 128, 01 2020. doi: 10.1007/s11263-019-01240-x.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Florian Bernard, Johan Thunberg, Paul Swoboda, and Christian Theobalt. Hippi: Higher-order projected power iterations for scalable multi-matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[5] Florian Bernard, Daniel Cremers, and Johan Thunberg. Sparse quadratic optimisation over the stiefel manifold with application to permutation synchronisation. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:238253392.

[6] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[7] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

[8] Luca Cosmo, Emanuele Rodolà, Andrea Albarelli, Facundo Mémoli, and Daniel Cremers. Consistent partial matching of shape collections via sparse modeling. *Computer Graphics Forum*, 36(1):209–221, 2017. doi: 10.1111/cgf.12796.

[9] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[10] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. March 2018.

[11] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. October 2021.

[12] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. doi: https://doi.org/10.1002/nav.3800030109.

[13] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proc. of ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 2020.

[14] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. February 2018.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, page 177–186, Goslar, DEU, 2013. Eurographics Association.

[17] Qixing Huang, Fan Wang, and Leonidas Guibas. Functional map networks for analyzing and exploring large shape collections. *ACM Trans. Graph.*, 33(4), jul 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601111. URL https://doi.org/10.1145/2601097.2601111.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[19] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. *The Eleventh International Conference on Learning Representations (ICLR)*, December 2022.

[20] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models, 2023.

[21] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. REPAIR: REnormalizing permuted activations for interpolation repair. In *The Eleventh International Conference on Learning Representations*, January 2023.

[22] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. May 2019.

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

[24] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[25] Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey, 2023.

[26] Ekdeep Singh Lubana, Eric J Bigelow, Robert P. Dick, David Krueger, and Hidenori Tanaka. Mechanistic mode connectivity. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 22965–23004. PMLR, 23–29 Jul 2023.

[27] Aviv Navon, Aviv Shamsian, Ethan Fetaya, Gal Chechik, Nadav Dym, and Haggai Maron. Equivariant deep weight space alignment, 2023.

[28] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/3df1d4b96d8976ff5986393e8767f5b2-Paper.pdf.

[29] Fidel A Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. Re-basin via implicit sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20237–20246, 2023.

[30] Xingyu Qu and Samuel Horvath. Rethinking model re-basin and linear mode connectivity, 2024. URL https://arxiv.org/abs/2402.05966.

[31] Frank R. Schmidt, Eno Töppe, Daniel Cremers, and Yuri Boykov. Intrinsic mean for semi-metrical shape retrieval via graph cuts. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, pages 446–455, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74936-3.

[32] Xinchu Shi, Haibin Ling, Weiming Hu, Junliang Xing, and Yanning Zhang. Tensor power iteration for multi-graph matching. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5062–5070, 2016. doi: 10.1109/CVPR.2016.547.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[34] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In Hugo Larochelle, Marc'aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[35] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21:343–348, 1967. URL https://api.semanticscholar.org/CorpusID:50329347.

[36] George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*.

[37] Paul Swoboda, Dagmar Kainmüller, Ashkan Mokarian, Christian Theobalt, and Florian Bernard. A convex relaxation for multi-graph matching. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11148–11157, 2019. doi: 10.1109/CVPR.2019.01141.

[38] Lanhui Wang and Amit Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference: A Journal of the IMA*, 2(2):145–193, 2013.

[39] Lirui Wang, Kaiqing Zhang, Allan Zhou, Max Simchowitz, and Russ Tedrake. Fleet policy learning via weight merging and an application to robotic tool-use, 2023.

[40] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *CoRR*, abs/1911.11308, 2019.

[41] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19908–19919. Curran Associates, Inc., 2020.

[42] Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. Disambiguating visual relations using loop constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1426–1433, 2010. doi: 10.1109/CVPR.2010.5539801.

[43] Allan Zhou, Kaien Yang, Kaylee Burns, Adriano Cardace, Yiding Jiang, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Permutation equivariant neural functionals, 2023.

[44] Zhanpeng Zhou, Yongyi Yang, Xiaojiang Yang, Junchi Yan, and Wei Hu. Going beyond linear mode connectivity: The layerwise linear feature connectivity, 2023.

Contents

A Additional details

Here we report in-depth explanations and additional experimental details. In particular, Appendix A.1 extensively outlines the most related works, Appendix A.2 shows the `Frank-Wolfe` algorithm for the pairwise case, while Appendix A.3 describes the `MergeMany` procedure presented in [1] for merging multiple models. We also report the `REPAIR` method in Appendix A.4. Finally, we show how the matching algorithm empirically converges in Appendix A.5.

A.1 Extended related work

We report here a thorough review of works that are relevant to our research, providing a comprehensive understanding of the context of our work.

Linear mode connectivity Mode connectivity is interested in modes, *i.e.* model parameters at convergence. In this regard, Frankle et al. [13] first studied the connectivity of the parameters of models that were trained for a few epochs from the same initialization, while Garipov et al. [14] investigated whether these can be connected through a high-accuracy path without requiring the same initialization. Simultaneously, Draxler et al. [10] proposed an algorithm to find a *Minimum Energy Path* (MEP) between two modes of a neural network, showing that these paths are mostly flat in both the training and test landscapes. This implies that many minima actually live in a shared low-loss valley rather than in distinct basins. On a different perspective, Zhou et al. [43] proposed to study a class of neural functionals which are permutation-equivariant by design. Recent research proposes to study model behavior in the weight space beyond linear mode connectivity: Lubana et al. [26] show that different "mechanisms" in related models prevent simple paths of low loss in the weight space, while Zhou et al. [44] studied the linear connections between the linear features of each layer of differently trained models.

Model merging Model merging [1, 29, 34, 20, 39, 36] has seen a surge of interest in the last years as a mean to ensemble models without incurring in the added computational cost. One of the first works in this direction is Singh and Jaggi [34], who proposed an optimal-transport based weight-matching procedure. Later, Ainsworth et al. [1] proposed three matching methods, one of which being data-free. Closer to our global optimization, Peña et al. [29] proposed a gradient-descent based procedure that iteratively updates soft permutation matrices maintaining their bistochasticity via a differentiable Sinkhorn routine. When the models to match have been trained on different tasks, Stoica et al. [36] introduce a more general "zip" operation that accounts for features that may be task-specific and further allow obtaining multi-headed models. Most recently, Navon et al. [27] proposed aligning models in the embedding space of a deep weight-space architecture. Finally, weight merging proved useful for large language models [20] and robotics [39]. For a complete survey of mode connectivity and model merging, we refer the reader to [25].

Cycle consistency Cycle consistency is a recurrent idea in computer vision and pattern recognition, where it appears under different names (e.g., "synchronization", "loop constraints", or "multi-way matching") depending on the task. In the area of multi-view 3D reconstruction, Zach et al. [42] were probably the first to make an explicit attempt at finding solutions meeting the cycle-consistency requirement, although without ensuring theoretical guarantees on the result. In geometry processing, Cosmo et al. [8] ensured cycle-consistent alignment of collections of 3D shapes using an n-fold extension of the Gromov-Wasserstein distance with sparsity constraints. Overall, cycle consistency is a recurring idea in the computer vision [38, 42, 2] graph matching [28, 37, 32] and geometry processing literature [17, 8, 4].

A.2 Pairwise Frank-Wolfe Algorithm

As introduced in Section 3, we optimize a layer-global objective by iteratively optimizing its linear approximation via the the Frank-Wolfe algorithm [12]. We compute the gradient of Equation (1) with respect to each permutation P_i, as the sum of two contributions for each ∇_{P_i}: one from permuting the rows of W_i and another from permuting the columns of W_{i+1}:

$$\nabla_{P_i} f = \underbrace{W_i^A P_{i-1} (W_i^B)^\top}_{\text{from permuting rows}} + \underbrace{(W_{i+1}^A)^\top P_{i+1} W_{i+1}^B}_{\text{from permuting columns}} . \tag{6}$$

We report in Algorithm 3 the Frank-Wolfe algorithm for the pairwise case.

Algorithm 3 Frank-Wolfe for pairwise Weight Matching

```
Require: Weights of two models $A$ and $B$ with $L$ layers, tolerance $\epsilon > 0$
Ensure: Approximate solution to Equation (1)
$\mathbf{P}^k \leftarrow$ identity matrices
repeat
    for $i = 1$ to $L$ do
        $P_i^k \leftarrow$ permutation acting on rows of $W_i$
        $P_{i-1}^k \leftarrow$ permutation acting on columns of $W_i$
        $\nabla_{P_i^K} f \mathrel{+}= W_i^A P_{i-1}^k (W_i^B)^\top$
        $\nabla_{P_{i-1}^k} f \mathrel{+}= (W_i^A)^\top P_i^k W_i^B$
    end for
    for $P_i^k \in \mathbf{P}^k$ do
        $\Pi_i \leftarrow \mathrm{LAP}(\nabla_{P_i^K} f)$
    end for
    $\alpha \leftarrow \textsc{LineSearch}(f, \mathbf{P}^k, \mathbf{\Pi})$
    for $P_i^k \in \mathbf{P}^k$ do
        $P_i^{k+1} = (1 - \alpha) P_i^k + \alpha\, \Pi_i$
    end for
until $\| f(A, B, \mathbf{P}^{k+1}) - f(A, B, \mathbf{P}^k) \| < \epsilon$
return $\mathbf{P}^k$
```

A.3 MergeMany Algorithm

Algorithm 4 reports the MergeMany procedure originally proposed by Ainsworth et al. [1] for merging multiple models, mainly consisting in alternating matching and aggregation until convergence. In practice, at each iteration, the procedure picks a reference model at random and matches all the other models to it. Then, they are all aggregated by averaging the weights.

Algorithm 4 MergeMany

```
Require: Model weights $\Theta_1, \dots, \Theta_N$
Ensure: A merged set of parameters $\tilde{\Theta}$.
repeat
    for $i \in \textsc{RandomPermutation}(1, \dots, N)$ do
        $\Theta' \leftarrow \frac{1}{N-1} \sum_{j \in \{1,\dots,N\} \setminus \{i\}} \Theta_j$
        $\pi \leftarrow \textsc{PermutationCoordinateDescent}(\Theta', \Theta_i)$
        $\Theta_i \leftarrow \pi(\Theta_i)$
    end for
until convergence
return $\frac{1}{N} \sum_{j=1}^{N} \Theta_j$
```

A.4 REPAIR

Observing a decay in the variance of the activations of the aggregated model, Jordan et al. [21] proposed `REPAIR`, which renormalizes the activations of the merged model to match the statistics of the original models. In particular, given two endpoint models with activations X_1 and X_2, the activations X_α of the interpolated model are renormalized to have statistics:

$$\mathbb{E}\left[X_\alpha\right] = (1 - \alpha) \cdot \mathbb{E}\left[X_1\right] + \alpha \cdot \mathbb{E}\left[X_2\right] \tag{7}$$

$$\mathrm{std}\left(X_\alpha\right) = (1 - \alpha) \cdot \mathrm{std}\left(X_1\right) + \alpha \cdot \mathrm{std}\left(X_2\right). \tag{8}$$

A.5 Convergence and efficiency

We report here the convergence of our matching algorithm. In particular, Figure 11 shows the objective values during the optimization, exhibiting the expected monotonic increase, while Figure 12 shows the step sizes result of the line search at each iteration. Interestingly, Figure 12a shows that the step sizes are generally decreasing, but descend in an alternating manner. This is likely due to the fact that the permutations are obtained as consecutive interpolations, where even steps result in a soft permutation matrix that is the average of the current and next permutation matrix, while odd steps generally result in a hard permutation matrix with entries in $[0, 1]$. Figure 13 finally shows the intermediate permutation values during the optimization: at each step, the entries of the permutation matrix are the linear interpolation of the current solution and the projected gradient with factor α given by the step size. The red values in the figure represent entries currently being updated, which are neither 1 (blue) or 0 (yellow).

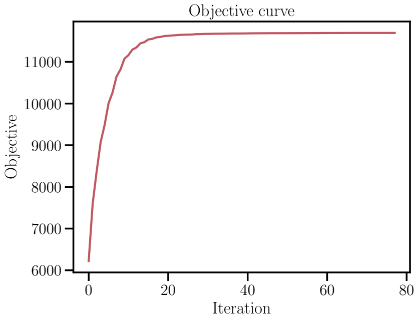


Figure 11: Objective values during the optimization. As guaranteed by the `Frank-Wolfe` algorithm, the objective value increases monotonically.

We report in Appendix A.5 the wall-clock time when merging $n = 2, 3$ `ResNet20` models having $1\times$, $2\times$, $4\times$, $8\times$ and $16\times$ width, together with their number of parameters.

	1x	2x	4x	8x	16x
# params	292k	1.166m	4.655m	18.600m	74.360m
			n=2		
C^2M^3	33.4s	33.5s	40.5s	80.8s	367.8s
`MergeMany`	0.24s	0.4s	3.4s	8.9s	59.4s
			n=3		
C^2M^3 time	32.9s	83.18s	91.0s	162.0s	715.8s
`MergeMany`	1.2s	4.1s	19.5s	105.8s	892.3s

Table 2: Wall-clock time for merging $n = 3$ ResNet20 models with different widths.

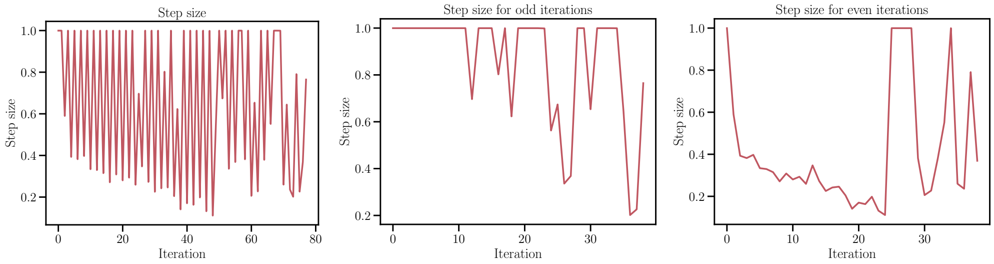
As can be inferred from the table, the scaling laws depend on the complexity of the resulting matching problem and cannot be predicted merely from the number of parameters, with a 4-fold increase in parameters resulting in no increase in runtime for the first three columns, a double increase in the second-last column and a 5-fold increase in the last. Compared to MergeMany, our approach enjoys a milder increase in running time when increasing the number of parameters. For simpler settings, however, `MergeMany` is significantly faster. Being the two approaches on the same order of magnitude and given the one-time nature of model merging, we believe this aspect to be of secondary importance, especially considering merging to be, in many cases, an alternative to training a model from scratch.

A.6 Architectural details

We report here the architectural details of all the architectures we have used in the experiments.

Multi-Layer Perceptrons We use a simple MLP mapping input to a 256-dimensional space followed by 3 hidden layers of 512, 512 and 256 units respectively, followed by an output layer mapping to the number of classes. We use *ReLU* activations for all layers except the output layer, where we use a *log_softmax* activation.

ResNet We consider a `ResNet20` [15] architecture composed by three ResNet block groups, each containing three residual blocks. The model starts with an initial convolutional layer followed by



(a) Step sizes for all iterations. (b) Step sizes for odd iterations. (c) Step sizes for even iterations.

Figure 12: Step sizes during the optimization.

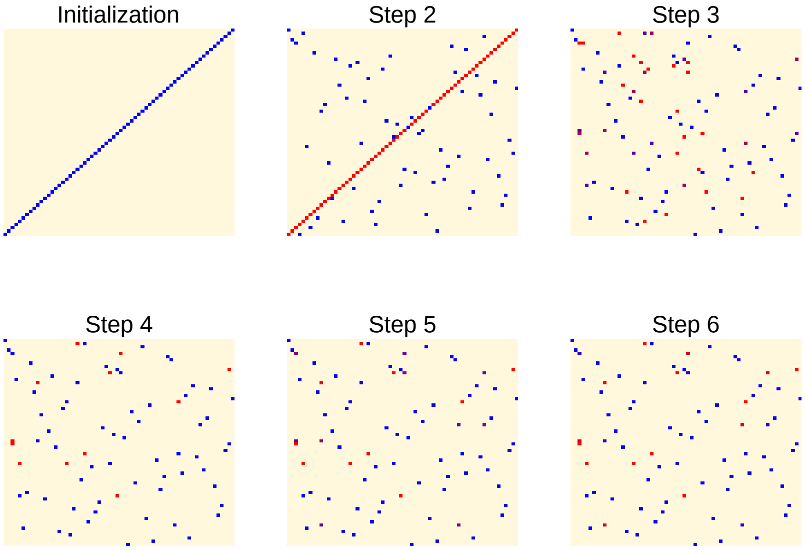


Figure 13: First 6 steps of Algorithm 3 for one permutation matrix. At each step, the new solution is given by the linear interpolation of the current solution and the gradient of Equation (1).

normalization and *ReLU* activation. It then passes through the three block groups with increasing channel sizes (determined by the widen factor) and varying strides, followed by global average pooling and a fully connected layer that outputs class logits. As normalization layers, we consider both the most commonly used *BatchNorm* [18] and, for the sake of comparing with `Git Re-Basin`, also *LayerNorm* [3]. The results in the main manuscript are all obtained with *LayerNorm*, while we report the results with *BatchNorm* in Appendix B.1.1.

VGG We employ a `VGG16` [33] architecture with *LayerNorm* [3] normalization layers. The model has the following convolutional layer dimensions, with “M” indicating the presence of a max-pooling layer

$$64, 64, M, 128, 128, M, 256, 256, 256, M, 512, 512, 512, M, 512, 512, 512, M \tag{9}$$

The convolutional layers are organized in 5 blocks, each containing 2 or 3 convolutional layers, followed by a max-pooling layer. The final classifier is composed of three fully connected layers with 512 hidden dimension and ReLU activations.

A.7 Datasets, hyperparameters and hardware details

We employ the most common datasets for image classification tasks: `MNIST` [9], `CIFAR-10` [23], `EMNIST` [7] and `CIFAR-100` [23], having 10, 10, 26 and 100 classes respectively. We use the standard train-test splits provided by *torchvision* for all datasets.

We use the same hyperparameters as `Git Re-Basin` where possible to ensure a fair comparison. In particular, we train most of our models with a batch size of 100 for 250 epochs, using SGD with momentum 0.9, a learning rate of 0.1, and a weight decay of 10^{-4}. We use a cosine annealing learning rate scheduler with a warm restart period of 10 epochs and a minimum learning rate of 0. We report each and every one of the hyperparameters used for each experiment, as well as all the trained models, in a WandB dashboard[2].

All of the experiments were carried out using consumer hardware, in particular mostly on a 32GiB RAM machine with a *12th Gen Intel(R) Core(TM) i7-12700F* processor and an *Nvidia RTX* 3090 GPU, except for some of the experiments that were carried on a 2080. Our modular and reusable codebase is based on *PyTorch*, leveraging *PyTorch Lightning* to ensure reproducible results and modularity and *NN-Template*[3] to easily bootstrap the project and enforce best practices.

A.8 Proofs

Theorem A.1. *The gradient of the objective function*

$$\sum_{p=1}^{n-1}\sum_{q=p+1}^{n}\sum_{\ell=1}^{L}\langle (P_\ell^p)^\top W_\ell^p P_{\ell-1}^p, (P_\ell^q)^\top W_\ell^q P_{\ell-1}^q\rangle$$

is Lipschitz continuous, implying our algorithm obtains a stationary point at a rate of $\mathcal{O}(1/\sqrt{t})$ [24].

Proof. We recall that, for each layer permutation $P^A = \{P_1^A, P_2^A, \dots, P_L^A\}$ of model A, we can define the gradient of our objective function relatively to the model B we are matching towards:

$$\begin{aligned} f(P_\ell^A) = \nabla_{P_\ell^A}^{\text{rows}} + \nabla_{P_\ell^A}^{\text{cols}} + \nabla_{P_\ell^A}^{\text{rows},\leftrightarrows} + \nabla_{P_\ell^A}^{\text{cols},\leftrightarrows} = \\ \left[W_\ell^A P_{\ell-1}^A (P_{\ell-1}^B)^\top (W_\ell^B)^\top + (W_{\ell+1}^A)^\top P_{\ell+1}^A (P_{\ell+1}^B)^\top W_{\ell+1}^B\right] P_\ell^B + \\ \left[W_\ell^B P_{\ell-1}^B (P_{\ell-1}^A)^\top (W_\ell^A)^\top + (W_{\ell+1}^B)^\top P_{\ell+1}^B (P_{\ell+1}^A)^\top W_{\ell+1}^A\right] P_\ell^A \end{aligned}$$

To prove Lipschitz continuity, we need to show there exists a constant C such that $\forall \ p = 1, \dots, n, \ \ \ell = 1, \dots, L \quad \|f(P_\ell^p) - f(Q_\ell^p)\| \leq C\|P_\ell^p - Q_\ell^p\|$. To simplify passages, we only consider a fixed ℓ and perform a generic analysis. We begin by observing that

$$\begin{aligned} f(P_\ell^p) - f(Q_\ell^p) = \\ \sum_{q\in[1,n]\setminus\{p\}} \left[W_\ell^p P_{\ell-1}^p (P_{\ell-1}^q)^\top (W_\ell^q)^\top + (W_{\ell+1}^p)^\top P_{\ell+1}^p (P_{\ell+1}^q)^\top W_{\ell+1}^q\right](P_\ell^q - Q_\ell^q) + \\ \left[W_\ell^q P_{\ell-1}^q (P_{\ell-1}^p)^\top (W_\ell^p)^\top + (W_{\ell+1}^q)^\top P_{\ell+1}^q (P_{\ell+1}^p)^\top W_{\ell+1}^p\right](P_\ell^p - Q_\ell^p) \end{aligned}$$

The last form of the above equation can be rewritten as a sum of the two sums:

$$\begin{aligned} \sum_{q\in[1,n]\setminus\{p\}} \left[W_\ell^p P_{\ell-1}^p (P_{\ell-1}^q)^\top (W_\ell^q)^\top + (W_{\ell+1}^p)^\top P_{\ell+1}^p (P_{\ell+1}^q)^\top W_{\ell+1}^q\right](P_\ell^q - Q_\ell^q) + \\ \sum_{q\in[1,n]\setminus\{p\}} \left[W_\ell^q P_{\ell-1}^q (P_{\ell-1}^p)^\top (W_\ell^p)^\top + (W_{\ell+1}^q)^\top P_{\ell+1}^q (P_{\ell+1}^p)^\top W_{\ell+1}^p\right](P_\ell^p - Q_\ell^p) \end{aligned}$$

Since the first term does not depend on either P_ℓ^p or Q_ℓ^p, we assume as a worst case that its norm is 0. Then, we remove transposes (since $\|M\| = \|M^\top\|$) and apply the triangle inequality and the sub-multiplicative property of matrix norms:

$$\begin{aligned} \|f(P_\ell^p) - f(Q_\ell^p)\| \leq \\ \sum_{q\in[1,n]\setminus\{p\}} \|P_\ell^p - Q_\ell^p\| \left(\|W_\ell^q\|\|P_{\ell-1}^q\|\|P_{\ell-1}^p\|\|W_\ell^p\| + \|W_{\ell+1}^q\|\|P_{\ell+1}^q\|\|P_{\ell+1}^p\|\|W_{\ell+1}^p\|\right) \end{aligned}$$

Let $C = \max_{q\in[1,n]\setminus\{p\}} \left\{\|W_\ell^q\|\|P_{\ell-1}^q\|\|P_{\ell-1}^p\|\|W_\ell^p\| + \|W_{\ell+1}^q\|\|P_{\ell+1}^q\|\|P_{\ell+1}^p\|\|W_{\ell+1}^p\|\right\}$. Then,

$$\|f(P_\ell^p) - f(Q_\ell^p)\| \leq C \sum_{q\in[1,n]\setminus\{p\}} \|P_\ell^p - Q_\ell^p\| = C(n-1)\|P_\ell^p - Q_\ell^p\|$$

[2] Link concealed to preserve anonymity.

[3] https://github.com/grok-ai/nn-template

Matcher	Barrier									
	ResNet$_{2\times}$		ResNet$_{4\times}$		ResNet$_{8\times}$		ResNet$_{16\times}$		VGG16	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Naive	5.16 ± 1.83	5.45 ± 1.83	2.94 ± 0.27	3.26 ± 0.27	2.12 ± 0.03	2.40 ± 0.03	1.84 ± 0.18	2.12 ± 0.17	1.85 ± 0.00	2.31 ± 0.00
Git Re-Basin	0.73 ± 0.16	0.86 ± 0.17	**0.74 ± 0.35**	0.80 ± 0.40	0.19 ± 0.03	0.13 ± 0.02	0.17 ± 0.02	0.07 ± 0.02	0.08 ± 0.03	0.24 ± 0.03
Frank-Wolfe	0.73 ± 0.19	0.85 ± 0.19	0.78 ± 0.33	0.81 ± 0.38	0.19 ± 0.03	0.12 ± 0.02	0.16 ± 0.02	0.06 ± 0.02	0.08 ± 0.03	0.25 ± 0.03

Table 4: Mean and standard deviation of the test and train loss barrier for each method when matching $n = 2$ models on CIFAR10.

we conclude that $f(P_\ell^p)$ is Lipschitz continuous for all models and all layers, with Lipschitz constant $C(n-1)$ depending on both the norm of the weights matrices and the number of models.

□

B Additional experiments

We report additional experiments and results in this section. In particular, Appendix B.1 presents a complete evaluation of our matching method for the pairwise case, showing it to be generally competitive with the state-of-the-art Git Re-Basin algorithm [1] and to outperform it on architectures employing *BatchNorm* [18] normalization. We then discuss different permutation initialization strategies in Appendix B.2.

B.1 Pair-wise model matching and merging

	Matcher	Barrier	
		Train	Test
ResNet$_{8\times}$	Naive	7.00 ± 1.24	8.37 ± 1.23
	Git-Rebasin	1.04 ± 0.10	1.54 ± 0.13
	Frank-Wolfe	**0.92 ± 0.06**	**1.42 ± 0.10**
VGG16	Naive	5.79 ± 0.39	7.36 ± 0.38
	Git-Rebasin	0.44 ± 0.03	0.64 ± 0.03
	Frank-Wolfe	**0.44 ± 0.05**	**0.63** ± 0.06

Table 3: Mean and standard deviation of the test and train loss barriers for each method when matching $n = 2$ models on CIFAR100.

As described in Section 3, our formalization can readily be used to match $n = 2$ models. In this case, the energy is given by Equation (1) and the permutations are not factorized. We compare the performance of our approach against the Git Re-Basin algorithm [1] and the naive baseline that aggregates the models by taking an unweighted mean on the original model weights without applying any permutation. From the data presented in Table 3, we observe that the approach is competitive with Git Re-Basin [1], with the two methods exhibiting analogously low test barrier on CIFAR10. Focusing on the ResNet20 architecture, we can see that width plays the same role in both approaches, with the barrier decreasing as it increases. We can also appreciate how, while the same architecture resulted in similar barriers for the two approaches on CIFAR10, the barrier is significantly lower for Frank-Wolfe in CIFAR100, possibly suggesting that the latter is more robust to the complexity of the dataset.

B.1.1 ResNet with BatchNorm

We also report the results of a ResNet20 with 2× width using *BatchNorm* [18] layers instead of *LayerNorm* [3] ones. This version, as noted in [21], is in fact harder to match but also the one that is commonly used in practice. We can see in Table 5 that the Frank-Wolfe matcher is able to achieve a lower barrier than Git Re-Basin, indicating the approach to be more robust to architectures using different normalization layers.

Matcher	loss barrier (↓)	
	train	test
Naive	4.72 ± 0.86	4.99 ± 0.86
Git Re-Basin	4.33 ± 0.64	4.62 ± 0.65
Frank-Wolfe	**3.53 ± 0.58**	**3.79 ± 0.57**

Table 5: Mean and stddev of the test and train loss barriers on 2 ResNet20-2× models with *BatchNorm* normalization.

B.2 Initialization strategies

As introduced in Algorithm 1, we initialize each N-dimensional permutation to be the $N \times N$ identity matrix. We now compare this strategy against two alternatives that provide doubly stochastic

models		1	2	3	4	5	6	7	8	9	mean	stddev	max gap	Frank-Wolfe
(1,2)	train	0.76	0.78	0.78	0.80	0.77	0.76	0.78	0.75	0.81	0.78	0.018	0.057	0.78
	test	0.73	0.75	0.75	0.77	0.74	0.73	0.74	0.72	0.78	0.74	0.018	0.060	**0.75**
(1,3)	train	0.67	0.69	0.69	0.69	0.62	0.69	0.66	0.71	0.68	0.68	0.023	0.085	0.68
	test	0.64	0.66	0.67	0.65	0.60	0.66	0.63	0.67	0.65	0.65	0.020	0.071	0.65
(2,3)	train	0.75	0.74	0.75	0.72	0.76	0.74	0.70	0.73	0.78	0.74	0.020	0.074	**0.76**
	test	0.70	0.71	0.71	0.68	0.72	0.70	0.67	0.70	0.74	0.70	0.020	0.071	**0.72**

Table 7: Accuracy of the interpolated model using `Git Re-Basin` [1] over different pairs of models $(1, 2), (1, 3), (2, 3)$ by changing random seed $i = 1, \ldots, 9$ in the weight matching procedure.

models	loss barrier (↓)		
	id	barycenter	Sinkhorn
(a, b)	0.52	**0.47**	0.60 ± 0.04
(b, c)	0.65	0.70	$\mathbf{0.64 \pm 0.06}$
(a, c)	0.97	0.95	$\mathbf{0.92 \pm 0.07}$

Table 6: Test barrier of the interpolations of 3 `ResNet20-2×` models using different initializations.

matrices, *i.e.*, such that their rows and columns sum to one: i) the Sinkhorn initialization [35] that initializes the permutation matrix as the solution of the Sinkhorn-Knopp algorithm [35]; ii) the barycenter of doubly stochastic matrices, *i.e.* the matrix where each element is given by $1/N$.Table 6 shows the test barrier of the interpolations of three `ResNet20-2×` models a, b, and c when using the different strategies over 10 different trials. We can see that the constant initializations (identity and barycenter) work well in general, with the additional benefit of having 0 variance in the results. On the other hand, if computational cost is not a concern, one can still choose to run a pool of trials with different Sinkhorn initializations and finally select the best one, trading this way efficiency with some extra accuracy points.

B.3 Variance of the results in Git Re-Basin

As introduced in Section 4, `Git Re-Basin` [1] depends on a random choice of layers, resulting in variations of up to 10% in accuracy depending on the optimization seed, while our method shows zero variance. While we have already seen the results for a model pair in Figure 4, we report, for completeness, the results of matching and averaging models with `Git Re-Basin` using different optimization seeds for additional pairs. As can be seen in Table 7, the trend is confirmed over these ones, with results significantly oscillating and our approach always above or on par with their mean.

B.4 Large-scale matching: ResNet50s trained over ImageNet

For this experiment, we matched three different `ResNet50`s trained over `ImageNet`. We used three publicly available pretrained checkpoints from *timm*, namely `a1_in1k`[4], `c1_1n1k`[5] and `ram_in1k` [6]. As Table 8 shows, C^2M^3 underperforms the baseline in this case. To see why, we report in Figure 14 the pairwise accuracies obtained using pairwise weight matching over all the `ResNet50` checkpoints available in *timm*. Let us focus on the triplet (am, a2, ram) and replace the model names with (a, b, c) for clarity. We see that, while the mergings (a, b) and (b, c) result in high-accuracy models, the merging (a, c) yields poor results. Given the cycle consistency of our method, we inherit the difficulty of the hardest pair, which in this case is (a, c). It is worth noting that this behavior is not present in the other cases we investigated in this work, and might be due to the considered models being trained with different training schedules and hyperparameters. Future research could investigate new strategies to handle such cases, *e.g.* by iteratively merging models by following a max-accuracy path in an accuracy weighted graph.

Matcher	**Accuracy** (↑)	**Loss** (↓)
`Naive`	0.001	6.91
`MergeMany`	0.001	6.91
`MergeMany`†	0.30	4.87
C^2M^3	0.001	6.91
$C^2M^{3\dagger}$	0.07	6.13

Table 8: Accuracy and loss of the interpolated model using different matchers over three `ResNet50` models trained on `ImageNet`.

[4] https://huggingface.co/timm/resnet50.a1_in1k

[5] https://huggingface.co/timm/resnet50.c1_in1k

[6] https://huggingface.co/timm/resnet50.ram_in1k

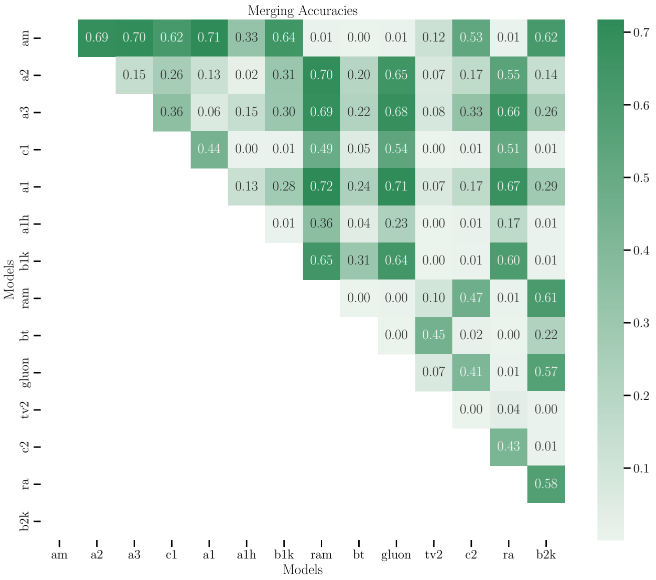


Figure 14: Pairwise accuracies obtained using `Git Re-Basin` [1] over different `ResNet50` models trained on `ImageNet`. Models are available from the *timm* library.

Round	Accuracy									
	1	5	10	15	20	25	30	35	40	45
FedAvg	0.0942	0.394	0.4972	0.5517	0.5699	0.5893	0.6018	0.6063	0.6099	0.6136
C^2M^3	0.0941	**0.4234**	**0.5193**	**0.5555**	**0.5783**	**0.5978**	**0.6077**	**0.6165**	**0.618**	**0.622**

Table 9: Accuracy over 10 clients in a federated learning scenario. We report the accuracy for 50 aggregation rounds, with each client training for 20 local epochs. We report one every five rounds for the sake of clarity.

B.5 Federated Learning

We here report the results of a preliminary experiment where we ran our framework in a federated learning scenario. To this end, we have used the state-of-the-art federated learning library Flower [7] [6] and employed our matching scheme over a set of 10 clients over `CIFAR10`, each adopting a small CNN model. We observe the following:

- When all the clients start from the same initialization, our approach has no benefit and falls back to standard averaging. In fact, the optimization process quickly returns identity matrices as permutations, suggesting the models already share the same basin.
- When instead we initialize the clients from different random initializations, Tables 9 and 10 show that our approach visibly outperforms FedAVG. In particular, the benefits get more pronounced when increasing the number of local epochs. This is in line with the intuition that standard averaging becomes less effective when clients drift due to prolonged local training and too infrequent aggregation.

While these results are not sufficient to claim an overall supremacy of the approach for the task due to the limited evaluation and choice of models, they show the approach to be promising for the problem and encourage further research.

[7] `https://flower.ai/`

Round	Accuracy									
	1	2	3	4	5	6	7	8	9	10
FedAvg	0.0942	0.2638	0.3543	0.3825	0.4165	0.4505	0.4742	0.4994	0.5169	0.5317
C^2M^3	**0.0947**	**0.3303**	**0.3899**	**0.4441**	**0.4764**	**0.4968**	**0.5184**	**0.5334**	**0.5434**	**0.5536**

Table 10: Accuracy over 10 clients in a federated learning scenario. We report the accuracy for 10 aggregation rounds, with each client training for 30 local epochs.

C Additional analysis

In this section, we report additional analyses that complement the results presented in the main text. We first analyze in Appendix C.1 how mapping to universe affects the similarity of the models; then, we evaluate how the composition of the match set affects the accuracy of the merged model in Appendix C.2.

C.1 Similarity of models

We analyze here how similar are models before and after being mapped to the universe space, first by comparing their representations and then by comparing their weights.

C.1.1 Representation-level similarity

Figures 15a and 15b show the Centered Kernel Alignment (CKA) [22] of the representations of 5 `ResNet20` models trained on `CIFAR10` with $2\times$ width. The linear version of CKA is defined as

$$\text{CKA}(X, Y) = \frac{\text{HSIC}(X, Y)}{\sqrt{\text{HSIC}(X, X)\,\text{HSIC}(Y, Y)}}, \tag{10}$$

where $\text{HSIC}(X, Y) = \frac{1}{(N-1)^2}\,\text{tr}(\mathbf{XHX}^\top\mathbf{H})$, $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{11}^\top$ is a centering matrix, and $\mathbf{1}$ is a vector of N ones. The denominator is introduced to scale CKA between zero and one, where a value of one indicates equivalent representations. CKA is invariant to orthogonal transformations and isotropic scaling. Being permutations orthogonal transformations, CKA stays exactly the same after mapping the models to the universe. On the contrary, the Euclidean distance of the representations of the models significantly decreases after mapping to the universe, as shown in Figures 15c and 15d.

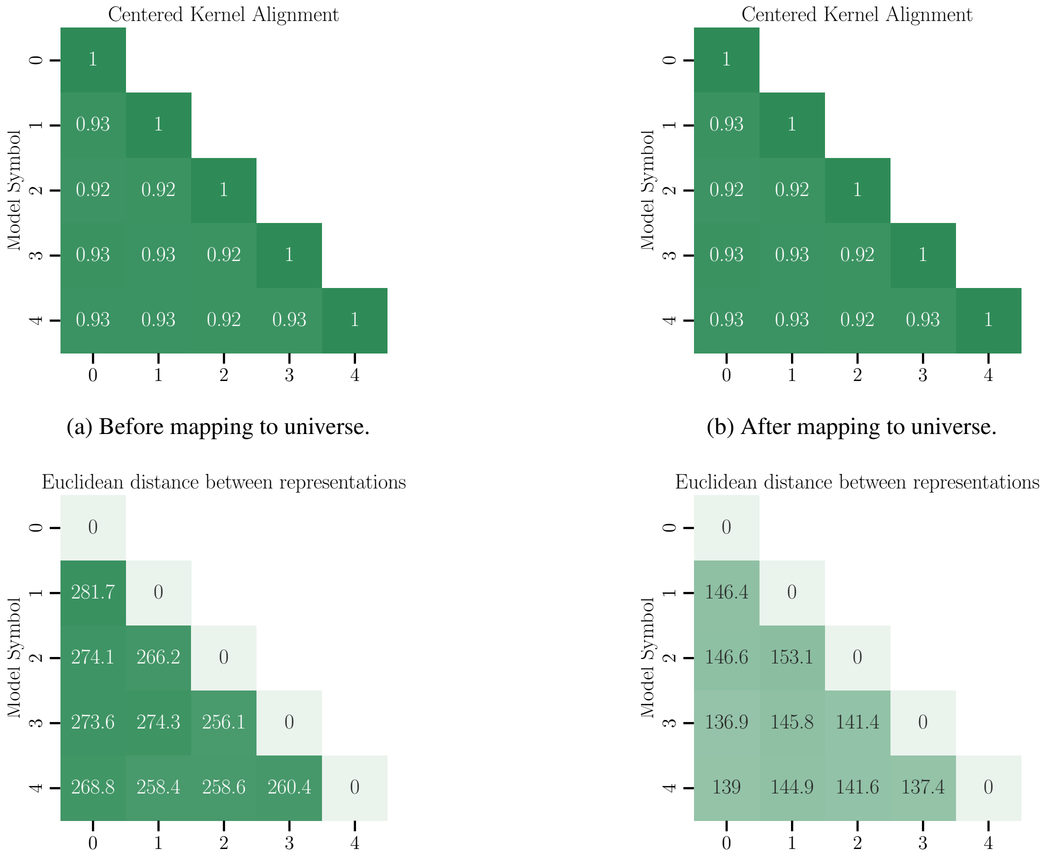
C.1.2 Weight-level similarity

We have seen in Figure 5 that the cosine similarity of the weights is higher after mapping the weights to the universe. This suggests that the models are more similar in the universe, which is consistent with the fact that it constitutes a convenient space to merge them. We report here for completeness the Figure 16 the Euclidean distance of the weights of 5 `ResNet20` models trained on `CIFAR10` with $2\times$ width, showing the same trend as the cosine similarity.

C.2 Merging different subsets

We merge subsets of $k < 5$ models from the set of 5 models a, b, c, d, e to gauge the effect of the match set composition over the accuracy of the merged model. As shown in Figure 17, we run two different merging schemes: in the former (left column), we globally match all the 5 models jointly and then consider subsets only at the aggregation step. In the second analysis (right column), we instead consider model subsets from the start, therefore running the whole matching procedure on the k models before averaging them. This way, we aim to disentangle the error resulting from imperfect matching from the one naturally resulting from the aggregation. We highlight a few notable aspects:

1. While the accuracies are expectedly higher when matching a subset with permutations expressly optimized for that same subset (right column), this is not the case for $n = 2$, in which the permutations resulting from matching the superset of 5 models yield better results when merging pairs of them. This hints at the added constraint of cycle consistency over a wide number of models adding in some cases an advisable prior over the search space.



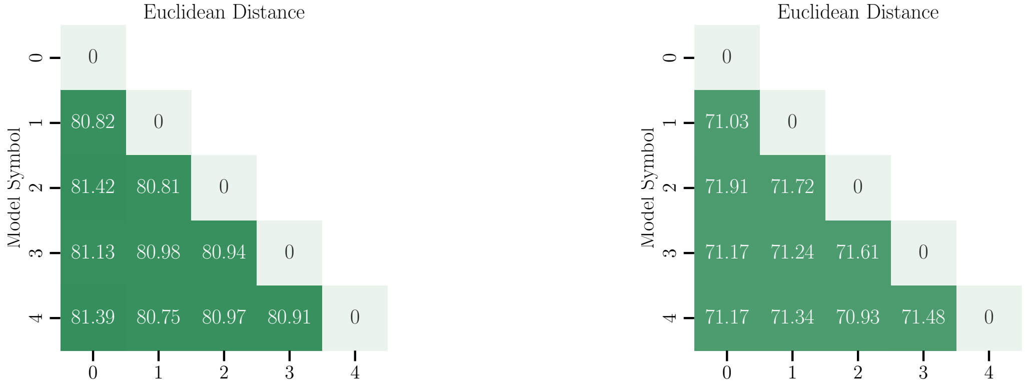
(a) Before mapping to universe.

(b) After mapping to universe.

(c) Before mapping to universe.

(d) After mapping to universe.

Figure 15: Cented Kernel Alignment and Euclidean distances of the representations of 5 `ResNet20` trained on `CIFAR10` with 2× width.



(a) Before mapping to universe.

(b) After mapping to universe.

Figure 16: Euclidean distance of the weights of 5 `ResNet20` trained on `CIFAR10` with 2× width.

2. The particular composition of the match set has a significant impact over the matching and subsequent merge operation, yielding differences of up to ≈ 20 accuracy points for the downstream model.
3. The standard deviations before the repair operation (red semi-transparent bars in the plots) are way lower when optimizing for the permutations over the superset of all 5 models; this suggests that the matching difficulty is spread over all the maps jointly, eventually yielding more stable results.

D Discussion

We discuss in this section the limitations of our work, as well as potential future societal impact.

D.1 On the cycle-consistency of C^2M^3

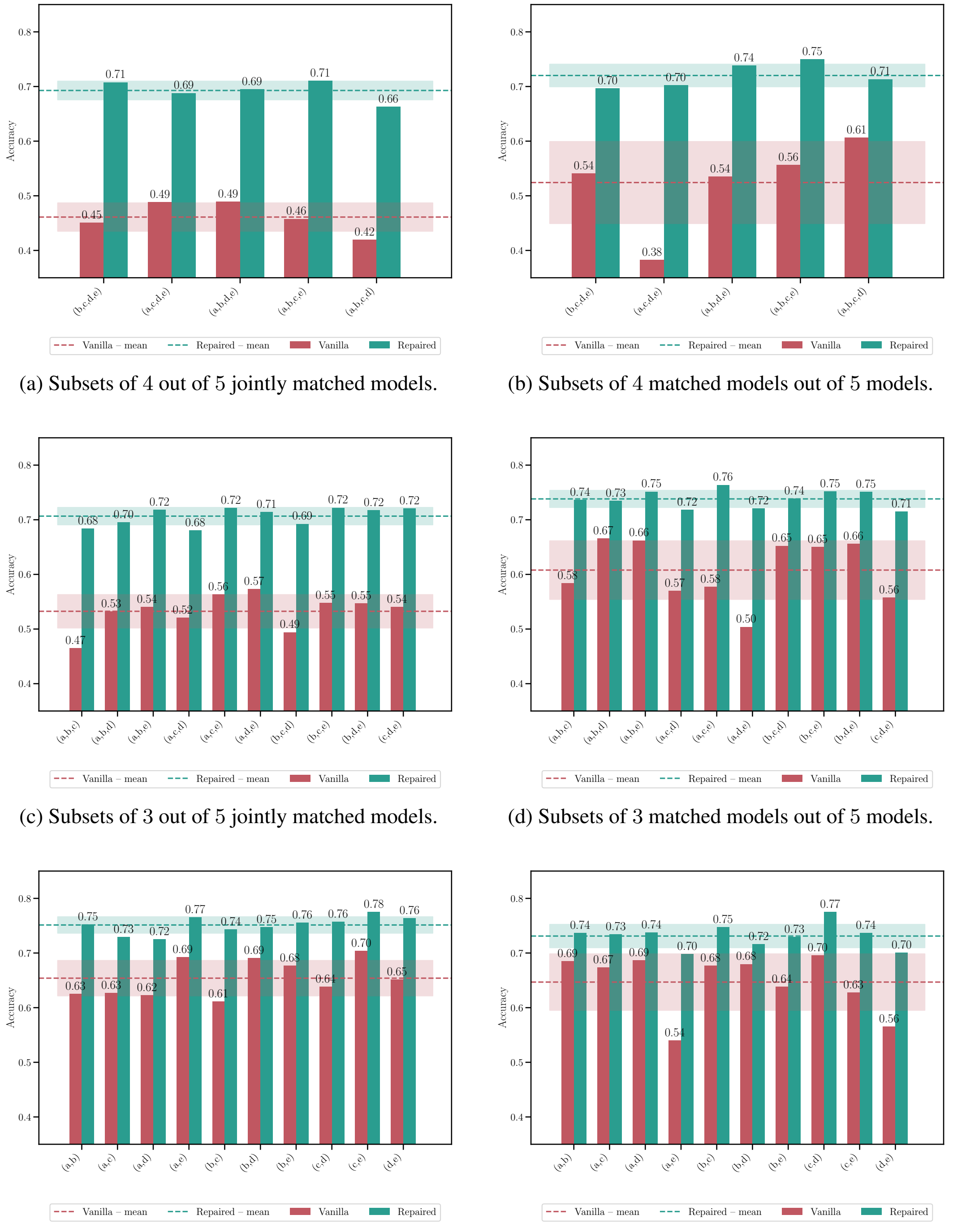
Our method is natively cycle-consistent due to the mathematical formulation of the optimization problem. If we were to not desire cycle consistency, the matching method would fall back to the $n = 2$ Frank-Wolfe (FW) case presented in Section 3. One would then have to transform the pairwise matching problem to a n-way matching problem, *e.g.* by using the $n = 2$ FW procedure as matching step in the `MergeMany` [1] algorithm. Results for the $n = 2$ FW matching are reported in Table 3.

D.2 Limitations

From what we have observed in our experiments, permutations satisfying linear mode connectivity of the models are hard to find for most architectures and datasets. In fact, given that there is no practical way to prove or disprove the conjecture for which most models end up in the same basin modulo permutations of the neurons, we cannot be sure that a certain set of models even allows finding such permutations, let alone that the permutations found are the optimal ones. We therefore encourage the community not to rely on the existence of such permutations in general. However, we have also shown that we can always find permutations that improve the resulting aggregated model, which is a promising practical result for model merging. As for all the existing works concerning linear mode connectivity and model merging, the resulting models that we obtain are sensible to a wide variety of factors, from training hyperparameters to the optimization algorithm used. Several works have already observed the phenomenon in practice: among these, Ainsworth et al. [1] mention among the known failure modes of their approaches models trained with SGD and too low learning rate, or ADAM coupled with too high learning rate. Jordan et al. [21] show that the chosen normalization layer incredibly affects the accuracy of the resulting merged model, while Qu and Horvath [30] observe learning rate, weight decay, and initialization method to play a strong role as well. Being a mostly empirical field, most of the technical choices that we make in our work mirror the ones made in previous works and are not based on a solid theoretical foundation. We therefore release all our code and encourage the community to investigate further on what training and optimization hyperparameters effect linear mode connectivity and model merging.

D.3 Societal impact and broader vision

The work presented in this paper serves as an additional tool for the community to improve the efficiency of deep learning models. By merging models, we can reduce the computational cost of training and inference, as well as the memory footprint of the models. In fact, by aggregating the information of a set of models into a single one with the same architecture, practitioners can benefit of the effects of ensembling without incurring in its computational cost. Moreover, merging is in many cases a practical necessity to guarantee confidentiality and privacy of user data, as it allows to train models on different subsets of the data, *e.g.* originating from different clients, and then merge them to obtain a single model integrating all the information. This is particularly important in the context of federated learning, where the data is distributed among different clients and cannot be shared. We believe that the work presented in this paper can be a stepping stone towards more efficient and privacy-preserving deep learning models, and we encourage the community to further investigate the potential of model merging in these contexts.



(a) Subsets of 4 out of 5 jointly matched models.

(b) Subsets of 4 matched models out of 5 models.

(c) Subsets of 3 out of 5 jointly matched models.

(d) Subsets of 3 matched models out of 5 models.

(e) Subsets of 2 out of 5 jointly matched models.

(f) Subsets of 2 matched models out of 5 models.

Figure 17: Accuracy of the resulting model when merging different model subsets. **(left)** performance of models obtained from aggregating subsets of $k < 5$ models that were matched jointly. **(right)** analoguous results for subsets of k models that are instead matched independently, *i.e.*, by only optimizing for the permutations that align those k models and discarding the remaining ones. The semi-transparent bands represent the standard deviation of the accuracy.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All the claims are supported by experiments and sound reasoning.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations and assumptions of our work in Appendix D.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper mostly leverages existing theoretical results and properly cites each of these ones.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe in great detail all of the presented algorithms and we report all the technical details for the architectures. We also provide a modular and reusable codebase that respects the highest software engineering standards with configuration parameters provided as separate yaml files. All the experiments were performed by setting reproducible seeds that are logged in WandB.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a modular and reusable codebase that respects the highest software engineering standards with configuration parameters provided as separate yaml files. Code is highly reproducible and machine agnostic due to the extensive use of frameworks and libraries such as PyTorch Lightning, WandB, Hydra and NN-template.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The most important details are provided in the paper, with the yaml configuration files in the code providing all the remaining minor details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report means and standard deviations for all our experiments when variance is present. The only entries that do not have standard deviation are those that result in a deterministic result that is not affected by random seed.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer ”Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: we reported in Appendix A the equipment used for the experiments, as well as a discussion on the efficiency of our methods.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn’t make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research totally complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss in Appendix D the potential positive societal impact of our work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We adequately cite all of the used datasets and architectures.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: While we provide a codebase that is extensible and modular enough to be reused by many researchers in the field, we are not releasing any libraries or datasets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.