FLOWMERGE: FLOW MATCHING FOR MODEL MERGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model merging is a recent area of study that attempts to find methods to merge individual models into a single checkpoint that retains the accuracies of its constituents. Most approaches to model merging rely on the Linear Mode Connectivity hypothesis, which posits that models trained under SGD lie in the same loss basin and can be merged by simply linearly interpolating through the checkpoints. In this paper, we propose that, rather than assuming a straight-line path in weight space, we learn a path in activation space using flow matching. We find that our method, **FlowMerge** outperforms Git Re-Basin on CIFAR-10 with MLP and ResNet-20 by 10 and 33 percentage points respectively. More broadly, our results suggest that model merging need not be confined to the linear interpolation paradigm implied by Linear Mode Connectivity, and that richer transport-based formulations may open up new possibilities for combining independently trained models.

1 INTRODUCTION

With an increase in the compute required for training stronger models, there has been a growing focus on ways to combine the knowledge of independently trained models into a single checkpoint, without incurring the cost of re-training them from scratch. This shows up in a variety of settings: federated learning, where models are trained by clients independently to alleviate privacy concerns, and then merged in a manner that doesn't require exchanging data or other identifying information (McMahan et al.). In multi-task training, models are individually trained on strong teacher checkpoints, and models need to be merged without losing accuracy on individual tasks. In continual learning settings, models need to be updated with new knowledge and abilities at regular intervals, without catastrophic forgetting (Kirkpatrick et al.). Such problems require taking two or more models and merging them into a single checkpoint, which allows us to define the problem of *Model Merging*.

There have been different approaches to model merging. However, a large number of these approaches rest on variations of the **Mode Connectivity** hypothesis (Garipov et al.; Draxler et al.). It postulates that different checkpoints of a model trained with SGD on the same dataset lie in a shared loss basin. It should be possible to merge them into a single checkpoint without affecting the loss too much. Simply averaging the weights of models results from this hypothesis and has been hugely popular.

Linear Mode Connectivity (LMC) (Frankle et al.) takes it a step further and proposes that such a step should be possible to accomplish using linear interpolation. Modern restatements of the hypothesis take into account the high permutation space that models form, positing that LMC holds modulo permutation, i.e. for every pair of models, there exists a permutation that transforms the models into a shared low-loss basin, through which we can linearly interpolate.[1] Git Re-Basin Ainsworth et al. uses this hypothesis to merge models by finding a permutation that minimizes the layerwise distance after permutation.

However, a linear path might be suboptimal in order to achieve high performance, particularly when finding the permutations is non-trivial. We take a step back and ask: *Can we find alternate, tractable methods to interpolate between the initial and final model*?

In this work, we answer this question affirmatively by drawing on **flow matching** (Lipman et al.), a modern generative modeling framework for learning continuous-time transport paths between two

[1]While there are other interpolation techniques like spherical interpolation, it has been found empirically that they perform mostly similar to linear interpolation, and hence we only refer to linear interpolation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

distributions. Rather than applying flow matching directly to the (very high-dimensional) space of network weights, we operate on the lower-dimensional space of logits induced by each model on a small calibration dataset. This lets us frame model merging as learning a vector field that transports the activation distribution of one model to that of another. For same-input endpoint pairs, we also bound the expected cross-entropy along the learned flow in terms of the endpoint risks and flow matching residual.

In our experiments, we identify that using an affine map to learn the flow performs nearly as well as using deeper models. Using this, we develop analytical solutions for learning the optimal parameters given some data. This allows us to not only reduce the time to estimate the flow matching head, but also results in more stable solutions. For a fixed Euler step sequence, the resulting affine transformation can be absorbed into the source classifier, preserving the discretized flow's predictions while reducing inference cost.

To summarize, our contributions are as follows:

- We introduce a framework called FlowMerge for model merging based on flow matching over logit distributions. It sidesteps the need to train a flow model directly in weight space.
- We study affine flow matching models, and identify analytical solutions for the optimal parameterization.
- We show empirically, on MLP and ResNet-20 architectures trained on CIFAR-10, that FlowMerge outperforms strong merging baselines, including Git Re-Basin and naive weight averaging, achieving nearly zero loss barriers between merged and source models.

2 BACKGROUND

Consider an L-layer multilayer perceptron (MLP) with hidden activations

$$\begin{aligned} \boldsymbol{h}_0 &= \boldsymbol{x}, \\ \boldsymbol{h}_l &= \sigma(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l), \qquad l = 1, \ldots, L-1, \\ f(\boldsymbol{x}; \boldsymbol{\theta}) &= \boldsymbol{W}_L \boldsymbol{h}_{L-1} + \boldsymbol{b}_L. \end{aligned} \tag{1}$$

Here σ acts elementwise, and $f(\boldsymbol{x}; \boldsymbol{\theta}) \in \mathbb{R}^{d_{\text{out}}}$ is the vector of logits. The parameters are fitted to a dataset $\mathcal{D}$ by minimizing a loss $\mathcal{L}(\boldsymbol{\theta})$, which is a proxy for the task to be achieved using the model.

Model merging The key question that model merging sets out to ask: given two independently trained checkpoints $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ of a model $\mathcal{M}$, can we combine them to get $\boldsymbol{\theta}_{\text{merged}}$, such that on both tasks, we obtain a low enough loss on the tasks both models were independently trained to solve. Therefore, under the formalism of the L-layer MLP defined earlier, LMC hypothesises that a merged checkpoint $\boldsymbol{\theta}^*_{\text{merged}}$ can be obtained simply by

$$\begin{aligned} \boldsymbol{W}_l^{\text{merged}} &= (1-\lambda)\boldsymbol{W}_l^{(0)} + \lambda \boldsymbol{W}_l^{(1)}, \\ \boldsymbol{b}_l^{\text{merged}} &= (1-\lambda)\boldsymbol{b}_l^{(0)} + \lambda \boldsymbol{b}_l^{(1)}, \qquad l = 1, \ldots, L, \end{aligned} \tag{2}$$

where $\lambda \in [0, 1]$ is the interpolation constant that determines the contribution of the individual models. This merged checkpoint will lie on a low loss basin characterized by the weight space of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$

Another variation of the LMC hypothesis attempts to account for hidden-unit permutation symmetries. Model B can first be aligned with model A by finding permutation matrices $\boldsymbol{P}_{l-1}^\top$ such that

$$\begin{aligned} \widetilde{\boldsymbol{W}}_l^{(1)} &= \boldsymbol{P}_l \boldsymbol{W}_l^{(1)} \boldsymbol{P}_{l-1}^\top, \\ \widetilde{\boldsymbol{b}}_l^{(1)} &= \boldsymbol{P}_l \boldsymbol{b}_l^{(1)}, \qquad l = 1, \ldots, L. \end{aligned} \tag{3}$$

Each $\boldsymbol{P}_l$ permutes the units in hidden layer l, while $\boldsymbol{P}_0$ and $\boldsymbol{P}_L$ are identity matrices so that input coordinates and class labels remain fixed. The aligned parameters then replace those of model B in the interpolation above.

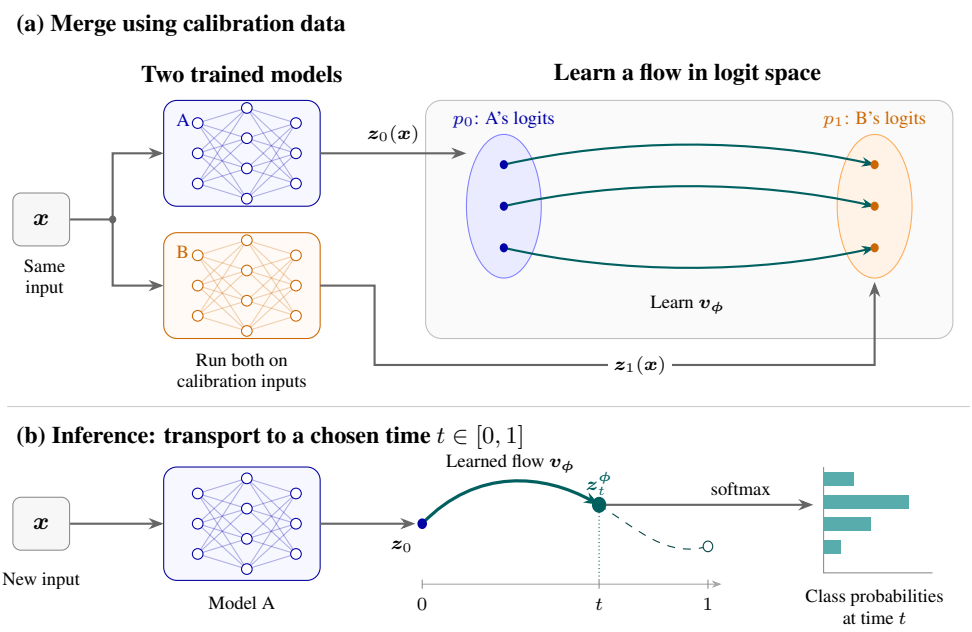

Figure 1: **FlowMerge training and inference.** (a) The same calibration input is passed through models A and B; their logit distributions provide endpoints for learning the vector field $\boldsymbol{v}_\phi$ (schematic). (b) At inference, model A's logits are transported along the learned flow to a chosen time t, then converted to class probabilities. The dashed curve shows the unused continuation toward time 1. For an affine head and fixed Euler steps to t, this transport can be folded into A's classifier to obtain a single checkpoint.

Flows and flow matching A flow describes a continuous transformation through the solution of an ordinary differential equation (ODE). Given a source distribution p_0 and a learned vector field $\boldsymbol{v}_\phi$, a trajectory satisfies

$$\frac{\mathrm{d}\boldsymbol{z}_t^\phi}{\mathrm{d}t} = \boldsymbol{v}_\phi(\boldsymbol{z}_t^\phi, t), \qquad \boldsymbol{z}_0^\phi = \boldsymbol{z}_0, \qquad \boldsymbol{z}_0 \sim p_0. \tag{4}$$

Flow matching trains this field using a prescribed path between endpoint samples. Let π be a coupling with source and target marginals p_0 and p_1. For $(\boldsymbol{z}_0, \boldsymbol{z}_1) \sim \pi$, the straight path $\bar{\boldsymbol{z}}_t = (1-t)\boldsymbol{z}_0 + t\boldsymbol{z}_1$ has constant velocity $\boldsymbol{z}_1 - \boldsymbol{z}_0$. The corresponding conditional flow matching objective is

$$\mathcal{L}_{\mathrm{FM}}(\phi) = \mathbb{E}_{\substack{(\boldsymbol{z}_0,\boldsymbol{z}_1)\sim\pi \\ t\sim\mathcal{U}[0,1]}} \left[\left\| \boldsymbol{v}_\phi(\bar{\boldsymbol{z}}_t, t) - (\boldsymbol{z}_1 - \boldsymbol{z}_0) \right\|_2^2 \right], \tag{5}$$

where $\mathcal{U}[0,1]$ denotes the uniform distribution and t is sampled independently of the endpoints. The learned trajectory $\boldsymbol{z}_t^\phi$ need not coincide with the straight training path $\bar{\boldsymbol{z}}_t$.

3 Method

It is not entirely straightforward to use flow matching to directly generate, or interpolate between, the weights of neural networks. This is due to a number of reasons, including but not limited to (a) requiring training a large number of models to be trained to obtain a distribution of weights, (b) requiring to take care of permutation invariance over the classes of neural networks and (c) training over the high-dimensional space of weights (in the multiple millions). This allows us to reduce our search space to a tractable dimensionality over which we can match flow.

We assume $\mathcal{M}$ to be a model defined as an L-layer MLP, with the final layer being a classifier head with the non-linearity being the softmax function. Furthermore, we also denote the input dimension of the model as d_{in}, and the output dimension as d_{out}, i.e. $\mathcal{M} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$. We denote the logits of the model as $\boldsymbol{z}_i(\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta}_i) \in \mathbb{R}^{d_{\text{out}}}$ for checkpoint $i \in \{0, 1\}$. Therefore, given n inputs $\boldsymbol{X} \in \mathbb{R}^{d_{\text{in}} \times n}$ stored as columns, we have two distributions of logits of the final layer

$\boldsymbol{Z}_0, \boldsymbol{Z}_1 \in \mathbb{R}^{d_{\text{out}} \times n}$. We can thus use these distributions to construct probability paths between them to use for flow matching.

During inference, we solve the ODE using the flows generated by flow matching head, and then use softmax to get the final probabilities of the model. Figure 1 summarizes the training and inference stages. Moving forward, we refer to the flow matching model used to interpolate between the distributions as the flow matching head, which precedes the flow matching model.

Importantly, we do not directly optimize the classification loss when training the flow matching head. However, we can show that if the vector field is Lipschitz and the learnt flow matching error is small, then the expected cross-entropy stays close to the linear interpolation of the endpoint risks.

Theorem 1 (Classification risk along the flow)**.** *Suppose the learned ODE has a solution on* $[0, 1]$ *for* $\mathbb{P}$*-almost every input, and* $\boldsymbol{v}_{\boldsymbol{\phi}}(\cdot, t)$ *is uniformly* K*-Lipschitz in the Euclidean norm. If the endpoint risks and* R_{FM} *are finite, then, for every* $t \in [0, 1]$*,*

$$R_{\text{CE}}(\boldsymbol{z}_t^{\boldsymbol{\phi}}; \mathbb{P}) \leq (1-t) R_{\text{CE}}(\boldsymbol{z}_0; \mathbb{P}) + t R_{\text{CE}}(\boldsymbol{z}_1; \mathbb{P}) + \sqrt{2}\, C_K(t) \sqrt{R_{\text{FM}}}, \tag{6}$$

where $C_K(t) = (\int_0^t e^{2K(t-s)}\, \mathrm{d}s)^{1/2}$.

Here $R_{\text{CE}}(\cdot; \mathbb{P})$ denotes cross-entropy risk under the input–label distribution $\mathbb{P}$, and R_{FM} is the expected integrated squared flow matching residual along same-input straight paths. The precise definitions and proof are given in Appendix A.1.

3.1 Affine maps for flow matching

In our experiments, a single affine layer provides a useful flow matching head. With $\boldsymbol{v}_{\boldsymbol{\phi}}(\boldsymbol{z}, t) = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}$, fitting the flow matching objective is a multivariate least-squares problem. Informally, the optimal affine head is the ordinary least-squares regression from intermediate activations to endpoint displacement.

Theorem 2 (Least-squares affine flow matching)**.** *Let* $\boldsymbol{Q}, \boldsymbol{D} \in \mathbb{R}^{d \times n}$ *contain intermediate logits and target displacements as columns, with column means* $\bar{\boldsymbol{q}}, \bar{\boldsymbol{d}}$ *and centered matrices* $\boldsymbol{Q}_{\text{c}}, \boldsymbol{D}_{\text{c}}$*. The affine field minimizing the empirical squared velocity error, with minimum Frobenius norm of* $\boldsymbol{W}$*, is*

$$\begin{aligned} \boldsymbol{W}^* &= \boldsymbol{D}_{\text{c}} \boldsymbol{Q}_{\text{c}}^\top (\boldsymbol{Q}_{\text{c}} \boldsymbol{Q}_{\text{c}}^\top)^\dagger, \\ \boldsymbol{b}^* &= \bar{\boldsymbol{d}} - \boldsymbol{W}^* \bar{\boldsymbol{q}}. \end{aligned} \tag{7}$$

Appendix A.2 provides a full proof.

Additionally, given a fixed time step sequence, we can convert flow matching models to residual networks. For affine heads, we can convert this to a single affine layer that can be fused with the existing model. This allows using the same model architecture, without requiring any additional layers or modules to be used in addition with the model.

4 Experiments

Baselines We evaluate our method against the four methods listed below. All approaches are parameterized by $\lambda \in [0, 1]$, with $\lambda = 0$ corresponding to model A and $\lambda = 1$ to model B for the interpolation baselines. For FlowMerge, λ is the ODE integration time; the learned endpoint only approximates model B.

- **Ensembling**: After passing the same input through both individual models, we average their output probabilities with weights $1 - \lambda$ for model A and λ for model B. We use this to indicate the *upper bound* of what is possible with model merging (aside from training from scratch). Note that an ensemble method requires both models to be used during inference, while model merging techniques aim to use strictly less compute than the ensemble.
- **Weight Averaging**: We average the checkpoints parameter-wise with weights $1 - \lambda$ for model A and λ for model B. This follows from the LMC hypothesis. We include this as it is a simple but effective strategy, used in a variety of circumstances.

Table 1: Results on the standard test split, evaluated at $\lambda = 0.5$, where $\lambda = 0$ is model A, and $\lambda = 1$ is model B. Bold and underlined means mark the best and second-best results amongst the model merging techniques. We observe that both the learnt and analytical variations of FlowMerge demonstrate SOTA performance for model merging. The differences become much more pronounced as we scale our dataset and model, with the other models unable to scale on standard-width ResNet-20s. (Ensembling is not strictly counted as a model merging technique, and is included here as a reference of the gold standard)

	MNIST MLP		**CIFAR-10** MLP		**CIFAR-10** ResNet-20		**CIFAR-100** ResNet-20		CIFAR-10 VGG-11
Method	Acc. (%) ↑	Loss ↓	Acc. (%) ↑	Loss ↓	Acc. (%) ↑	Loss ↓	Acc. (%) ↑	Loss ↓	Acc. (%) ↑
Model A	98.55 ±0.12	0.1223 ±0.0169	53.69 ±0.26	1.3766 ±0.0256	87.53 ±0.23	0.3917 ±0.0056	57.62 ±0.27	1.6082 ±0.0076	89.77
Model B	98.65 ±0.09	0.1217 ±0.0172	53.74 ±0.29	1.3624 ±0.0226	87.71 ±0.36	0.3829 ±0.0072	57.59 ±0.28	1.6148 ±0.0016	80.12
Ensemble	98.73 ±0.07	0.0865 ±0.0054	57.07 ±0.24	1.2286 ±0.0084	88.99 ±0.16	0.3320 ±0.0049	60.53 ±0.24	1.4489 ±0.0057	86.28
Weight averaging	83.89 ±10.23	0.8100 ±0.1890	11.05 ±0.72	3.0216 ±0.3445	11.46 ±2.76	3.7837 ±0.9039	1.24 ±0.20	8.9609 ±1.4664	10.00
Git Re-Basin (Activation)	98.18 ±0.08	0.1123 ±0.0090	48.96 ±1.16	<u>1.4531</u> ±0.0274	48.45 ±6.05	1.7759 ±0.3107	14.89 ±3.08	4.5910 ±0.5222	11.73
Git Re-Basin (Weight)	<u>98.38</u> ±0.08	**0.0886** ±0.0078	39.99 ±3.42	1.6474 ±0.0884	42.69 ±6.14	1.9098 ±0.2845	11.49 ±3.82	5.1596 ±0.8211	<u>10.00</u> ±
ZipIt	98.21 ±0.09	<u>0.1097</u> ±0.0090	46.43 ±3.22	1.5016 ±0.0547	16.91 ±5.17	4.4215 ±2.5875	2.37 ±1.08	8.6141 ±1.0613	–
FlowMerge (ours)	98.26 ±0.21	0.1429 ±0.0233	<u>50.42</u> ±1.07	1.4929 ±0.0266	<u>82.79</u> ±0.31	<u>0.5721</u> ±0.0083	<u>47.98</u> ±0.33	<u>2.1467</u> ±0.0179	**83.27** ±
Analytical FlowMerge (ours)	**98.52** ±0.09	0.1214 ±0.0103	**53.64** ±0.58	**1.3779** ±0.0283	**87.53** ±0.20	**0.3905** ±0.0060	**57.04** ±0.40	**1.6196** ±0.0096	–

- **Activation and Weight Matching**: Ainsworth et al. proposes two approaches, weight and activation matching, to find permutations that can be used. Git Re-Basin canonically refers to the weight matching procedure, which finds permutations by maximizing the similarity in weights. Since we require a calibration budget, we also compare against activation matching, which finds permutations by maximizing the similarity in activations.
- **ZipIt!**: Stoica et al. (2024) propose merging models by merging redundant features (both across different models, as well as within the same model). Since ZipIt! does not have an interpolation variable λ, we merely report the result after merging all layers.

4.1 Merging independently trained models

We evaluate models trained on the MNIST (Lecun et al., 1998) and CIFAR-10 (Krizhevsky, 2009) datasets with different hyperparameters. The results of our experiments are presented in Table 1. We find that FlowMerge outperforms all approaches, performing better than individual models and approaching the performance of the ensemble, while using half the number of parameters. On a four-layer MLP trained on MNIST, all approaches have a similar performance of 98%, except weight averaging.

Using MLPs trained on CIFAR-10, we find that the activation matching procedure, despite having access to a calibration set of samples, is unable to match the performance of FlowMerge. FlowMerge, on the other hand, is able to approach the performance of the ensemble, and within the error bounds of Git Re-Basin. On training CIFAR-10 with ResNet-20 He et al. (2016), FlowMerge is the only approach that results in a low loss barrier (0.5721 $\pm$0.0083, compared to 3.78 $\pm$0.9 for weight averaging and 1.78 $\pm$0.3 for weight matching). This is, to the authors' knowledge, the first approach that results in a near-zero loss barrier for the ResNet-20 architecture at its original width.

4.2 Merging models trained on disjoint datasets

Another common setting for evaluating model merging is by splitting a single dataset into two splits A and B, obtaining two checkpoints $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ that are trained on each split respectively, and merging them. We perform the same experiment with ResNet-20 on CIFAR-10. To construct the dataset splits, we first partition the dataset into two disjoint sets of classes. Split A is then formed by sampling 80% of the first set, and 20% of the second set, and vice-versa for B.

The results of merging two ResNet-20 models on CIFAR-100 are presented in Table 2. At $\lambda = 0.5$, FlowMerge is able to neatly interpolate between the accuracies (and losses) of individual models, while keeping the joint accuracy higher than individual models. This demonstrates that the model is not biased towards either split, it is able to learn how to interpolate without undergoing catastrophic changes in accuracy or loss.

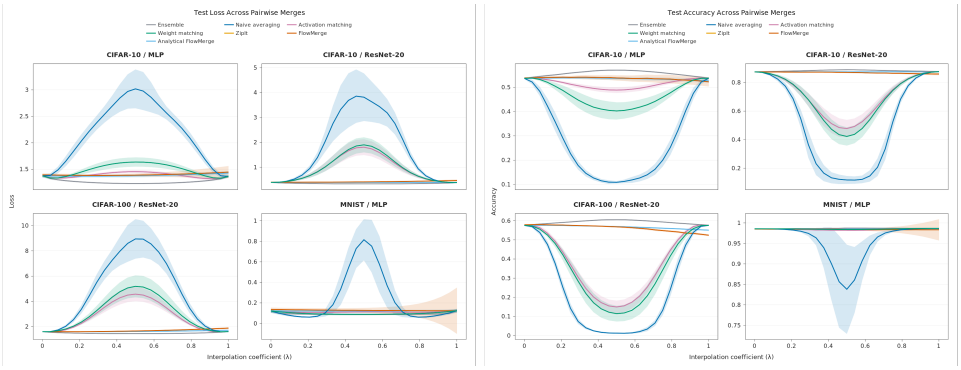

Figure 2: Loss and accuracy for models after merging on CIFAR-10. We notice that both the analytical and learnt solutions for FlowMerge perform nearly identical to the ensemble, and far outperform any other model merging technique.

Table 2: Accuracy and loss on the test set of CIFAR-100, using Resnet-20 trained on the train set. All results are reported at $\lambda = 0.5$. Bold and underlined means mark the best and second-best results among the bottom four methods for each test set and metric. Both variations of FlowMerge are able to interpolate between the accuracies (and thus, predictions) of the two models, which implies that the flow matching head is able to learn effective flows that direct the induced probability distribution over time.

	Test Split 1		Test Split 2		Full test set	
Method	Acc. (%) ↑	Loss ↓	Acc. (%) ↑	Loss ↓	Acc. (%) ↑	Loss ↓
Model A (Split 1)	31.65 ±0.49	2.9575 ±0.0110	60.18 ±0.76	1.5627 ±0.0255	45.91 ±0.55	2.2601 ±0.0124
Model B (Split 2)	56.66 ±0.32	1.6994 ±0.0189	31.34 ±0.55	2.9544 ±0.0554	44.00 ±0.31	2.3269 ±0.0309
Ensemble	51.62 ±0.44	1.8711 ±0.0151	54.45 ±0.76	1.7779 ±0.0217	53.04 ±0.40	1.8245 ±0.0098
Weight averaging	1.63 ±1.02	7.5682 ±1.5526	1.09 ±0.99	7.6884 ±1.6492	1.36 ±0.28	7.6283 ±1.5840
Git Re-Basin (Activation)	15.69 ±2.78	4.4416 ±0.5478	16.23 ±3.31	4.2369 ±0.4261	15.96 ±2.87	4.3392 ±0.4833
Git Re-Basin (Weight)	12.34 ±2.97	4.6431 ±0.3999	12.75 ±3.44	4.6284 ±0.5208	12.55 ±3.09	4.6357 ±0.4279
ZipIt	3.26 ±1.01	7.8423 ±1.3594	3.49 ±1.07	7.3781 ±0.9339	3.37 ±0.84	7.6102 ±1.1127
FlowMerge (ours)	<u>39.92</u> ±0.23	<u>3.2734</u> ±0.0478	<u>46.96</u> ±0.68	<u>2.2832</u> ±0.0458	<u>43.44</u> ±0.41	<u>2.7783</u> ±0.0429
Analytical FlowMerge (ours)	**40.01** ±0.35	**2.3659** ±0.0116	**52.69** ±0.54	**1.8258** ±0.0114	**46.35** ±0.35	**2.0958** ±0.0086

4.3 Varying the depth of the flow matching head

There are two main reasons why we wish to compare the performance of the flow matching head at varying depths:

- While we want the flow matching model to learn a mapping between the two models, it shouldn't be possible that the flow matching model uses its own parameterization to directly improve the accuracy of the model. In other words, a deeper model should ideally not result in an accuracy greater than that of the original model after performing merging.
- An affine flow matching head (i.e. only a single layer) allows us to make additional optimizations in both training and inference.

To test this, we ablate the depth of the flow matching head, varying from a single affine layer (depth 0) to a model of depth 8. We find that the model is largely agnostic of depth, demonstrating that the performance is largely through the inherent modeling of the initial checkpoints and not the flow-matching procedure. Furthermore, we find that a single affine layer reaches an admirable performance of 44.42% against 57.02% for deeper layers, when used to merge ResNets trained on CIFAR-10. These results show that even an affine layer is sufficient for learning the flow between the models, and additional layers do not bring a meaningful difference to the model's accuracy.

Table 3: Effect of flow depth when merging CIFAR-100 ResNet-20 models trained on separate data splits. Full test-set accuracy and loss are shown at two interpolation points as mean ± sample standard deviation across five seeds. We observe that the performance caps off at a depth of 4 layers.

	$\lambda = 0.5$		$\lambda = 1.0$	
Flow depth	Accuracy (%)	Loss	Accuracy (%)	Loss
0	44.42 ± 0.41	2.278 ± 0.043	39.18 ± 0.69	2.545 ± 0.066
1	46.23 ± 0.59	2.210 ± 0.022	41.57 ± 0.77	2.479 ± 0.034
2	46.83 ± 0.48	2.165 ± 0.014	44.27 ± 0.34	2.301 ± 0.023
4	$\mathbf{47.20 \pm 0.44}$	$\mathbf{2.146 \pm 0.013}$	$\mathbf{44.84 \pm 0.49}$	$\mathbf{2.262 \pm 0.010}$
8	46.93 ± 0.64	2.164 ± 0.016	44.73 ± 0.86	2.271 ± 0.041

When we compare the results of calculating the flow analytically against learning the flow, we find that the analytic calculation generally has a higher accuracy and a lower loss compared to the other methods. We notice that it tends to model the flow in a cleaner manner, allowing for better accuracy.

5 Related Work

5.1 Model Merging

Mode connectivity describes low-loss paths between solutions in neural-network loss landscapes (Garipov et al.; Draxler et al.). Frankle et al. study linear mode connectivity for models trained from a shared initialization under different sources of stochasticity. For independently initialized models, permutation symmetries of hidden units can explain some of the barriers to direct weight interpolation; Entezari et al. (2022) study this connection, and Git Re-Basin finds permutations that align units before averaging or interpolating the weights (Ainsworth et al.). These results motivate parameter-space merging, while also showing that successful interpolation depends on the relationship between the models and the alignment used.

Several methods explicitly align models before merging. Singh and Jaggi (2020) use optimal transport to softly align neurons before parameter averaging. MuDSC (Xu et al., 2024) combines weight- and activation-based similarities when selecting permutations, including for models with group structure. ZipIt! (Stoica et al., 2024) merges features both across and within models, and can leave later layers unmerged when combining models trained on different tasks. Theus et al. (2025) broaden the symmetry-aware alignment introduced in Ainsworth et al. beyond permutations and study connectivity in transformers. Adilova et al. (2024) analyze when averaging individual layers or groups of layers preserves low loss, and whether individual layers lie in a shared low-loss basin.

The closest output-space formulation is the recent preprint by Evans et al. (2026), which chooses combinations of fine-tuning updates by minimizing a squared-output calibration objective under a linearized model. FlowMerge instead learns a vector field between the logit distributions of two classifiers, yielding a continuous path in activation space rather than selecting a fixed combination of parameter updates.

5.2 Flow Matching

Flow matching (FM) is a simulation-free way to train continuous normalizing flows (CNFs). Rather than fitting a transformation by repeatedly solving its ODE during training, FM specifies a probability path between a source and target distribution and regresses a neural vector field to the path's known velocity. Once trained, the vector field defines an ODE whose numerical solution transports samples along the learned path. The choice of probability path is flexible: linear interpolation is one option, while other paths, including optimal-transport displacement paths, can be used when their velocities are tractable to compute (Lipman et al.).

Conditional flow matching (CFM) makes this regression objective practical by conditioning on endpoint samples, while preserving the desired marginal probability path. Its optimal-transport variant, OT-CFM, changes how endpoints are coupled and can produce simpler flows that are faster to

integrate (Tong et al.). This coupling matters because independently pairing source and target samples can create unnecessarily long or intersecting trajectories. Multisample flow matching addresses this by using minibatch couplings, such as minibatch optimal transport, to reduce gradient variance and encourage straighter learned trajectories, which can lower the number of function evaluations needed at inference (Pooladian et al.). Rectified Flow also studies straightening transport trajectories and reducing integration steps through iterative rectification (Liu et al., 2023).

FlowMerge applies these ideas to model merging rather than data generation. Its source and target distributions are the activations induced by two classifiers on a calibration set, and the endpoint coupling determines which activations the flow is trained to connect.

6 Conclusion

We introduce a novel way of model merging using flow matching to interpolate in the logit space. Our experiments show that this approach can improve on parameter-space merging baselines like Git Re-Basin and ZipIt!, without significantly increasing the compute required. We also derive closed-form fits for affine flow matching heads, whose discretized transformations can be absorbed into the parent model to ensure that we do not .

Further work could use advances in flow matching to improve both training and inference efficiency.

6.1 Limitations

Requirement of a calibration dataset Our method currently requires a calibration dataset to train the flow-matching model. This might not always be present to use, for example, in cases where privacy overrules the performance gain. Additionally, reliance on a calibration dataset does not provide adequate insight into the underlying weight symmetries and dynamics.

Speed Our implementation is significantly slower than Git Re-Basin, making it impractical for use cases such as federated learning, where independently trained models are merged into a single checkpoint to avoid leaking data. We leave it as open work for future work to incorporate ideas from the broader flow matching literature to speed both training and inference, including techniques used for single-step flow generation.

AI use statement

In this work, we used generative AI tools to assist with experiments and adapting reproductions of existing methods. While doing so, we carefully check the outputs of the tools to ensure that they are faithful to the methods and experiments intended to be run by the authors. We also used generative AI to assist with rewriting proofs and standardizing notation in the manuscript, as well as filling in gaps of literature that the authors were unfamiliar with. However, we have not used generative AI to develop our ideas or otherwise shape it.

References

Linara Adilova, Maksym Andriushchenko, Michael Kamp, Asja Fischer, and Martin Jaggi. Layer-wise linear mode connectivity. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LfmZh91tDI.

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1309–1318. PMLR. URL https://proceedings.mlr.press/v80/draxler18a.html.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *ICLR*, 2022. URL https://arxiv.org/abs/2110.06296.

Bethan Evans, Benjamin Etheridge, Stephen Roberts, and Jared Tanner. Model merging by output-space projection, 2026. URL https://arxiv.org/abs/2605.29101.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3259–3269. PMLR. URL https://proceedings.mlr.press/v119/frankle20a.html.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8803–8812. Curran Associates Inc. URL https://dl.acm.org/doi/10.5555/3327546.3327556.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition . In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, June 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.90. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. 114(13):3521–3526. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114. URL https://pnas.org/doi/full/10.1073/pnas.1611835114.

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

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282. PMLR. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: straightening flows with minibatch couplings. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pages 28100–28127. JMLR.org.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NeurIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

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*, 2024. URL https://openreview.net/forum?id=LEYUkvdUhq.

Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 136197–136225. Curran Associates, Inc., 2025. doi: 10.52202/

085713-4546. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c6eadcc507edc04c1baf00f05cd18b1a-Paper-Conference.pdf.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW.

Zhengqi Xu, Ke Yuan, Huiqiong Wang, Yong Wang, Mingli Song, and Jie Song. Training-free pretrained model merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5915–5925, June 2024.

A Theoretical Guarantees

A.1 Expected classification loss along a learned flow

Let $\mathbb{P}$ be a joint distribution over inputs $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$ and class labels $y \in \{1, \dots, d_{\text{out}}\}$. For a logit vector $\boldsymbol{z}$, write

$$\ell_{\text{CE}}(\boldsymbol{z}, y) = -\log \text{softmax}(\boldsymbol{z})_y. \tag{8}$$

The expected cross-entropy risk of a logit-valued function $\boldsymbol{g}$ is

$$R_{\text{CE}}(\boldsymbol{g}; \mathbb{P}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{P}}\big[\ell_{\text{CE}}(\boldsymbol{g}(\boldsymbol{x}), y)\big]. \tag{9}$$

For the two checkpoints, let $\boldsymbol{z}_i(\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta}_i)$, $i \in \{0, 1\}$. We compare their straight logit path

$$\bar{\boldsymbol{z}}_t(\boldsymbol{x}) = (1-t)\boldsymbol{z}_0(\boldsymbol{x}) + t\boldsymbol{z}_1(\boldsymbol{x}), \qquad t \in [0, 1], \tag{10}$$

with the learned trajectory starting from the same source logits:

$$\frac{\mathrm{d}\boldsymbol{z}_t^{\boldsymbol{\phi}}(\boldsymbol{x})}{\mathrm{d}t} = \boldsymbol{v}_{\boldsymbol{\phi}}(\boldsymbol{z}_t^{\boldsymbol{\phi}}(\boldsymbol{x}), t), \qquad \boldsymbol{z}_0^{\boldsymbol{\phi}}(\boldsymbol{x}) = \boldsymbol{z}_0(\boldsymbol{x}). \tag{11}$$

The relevant flow matching residual is measured along straight paths whose endpoints come from the same input:

$$R_{\text{FM}} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{P}}\left[\int_0^1 \|\boldsymbol{v}_{\boldsymbol{\phi}}(\bar{\boldsymbol{z}}_s(\boldsymbol{x}), s) - (\boldsymbol{z}_1(\boldsymbol{x}) - \boldsymbol{z}_0(\boldsymbol{x}))\|_2^2 \, \mathrm{d}s\right]. \tag{12}$$

The constant in Theorem 1 evaluates to

$$C_K(t) = \begin{cases} \sqrt{\dfrac{e^{2Kt}-1}{2K}}, & K > 0, \\ \sqrt{t}, & K = 0. \end{cases} \tag{13}$$

In particular, if both endpoint risks are at most ϵ, then $R_{\text{CE}}(\boldsymbol{z}_t^{\boldsymbol{\phi}}; \mathbb{P}) \le \epsilon + \sqrt{2}\, C_K(t)\sqrt{R_{\text{FM}}}$.

Proof of Theorem 1. The argument has two parts. First, we use the flow matching residual to bound how far the learned trajectory can move from the straight path. We then translate this distance into a cross-entropy bound.

Fix an input $\boldsymbol{x}$ and suppress its dependence in the notation. Let $\boldsymbol{d} = \boldsymbol{z}_1 - \boldsymbol{z}_0$ be the endpoint displacement, and define

$$\boldsymbol{r}_s = \boldsymbol{v}_{\boldsymbol{\phi}}(\bar{\boldsymbol{z}}_s, s) - \boldsymbol{d}, \qquad \boldsymbol{e}_s = \boldsymbol{z}_s^{\boldsymbol{\phi}} - \bar{\boldsymbol{z}}_s. \tag{14}$$

Thus $\boldsymbol{r}_s$ is the velocity residual along the straight path, while $\boldsymbol{e}_s$ is the difference between the two paths. Since the straight path has velocity $\boldsymbol{d}$ and both paths start at $\boldsymbol{z}_0$, we have

$$\dot{\boldsymbol{e}}_s = \boldsymbol{v}_{\boldsymbol{\phi}}(\boldsymbol{z}_s^{\boldsymbol{\phi}}, s) - \boldsymbol{v}_{\boldsymbol{\phi}}(\bar{\boldsymbol{z}}_s, s) + \boldsymbol{r}_s, \qquad \boldsymbol{e}_0 = \boldsymbol{0}. \tag{15}$$

The Lipschitz assumption bounds the first difference by $K\|\boldsymbol{e}_s\|_2$. Integrating and applying Grönwall's inequality therefore gives

$$\|\boldsymbol{e}_t\|_2 \le \int_0^t e^{K(t-s)}\|\boldsymbol{r}_s\|_2 \, \mathrm{d}s. \tag{16}$$

By Cauchy–Schwarz,

$$\|\boldsymbol{e}_t\|_2^2 \le \left(\int_0^t e^{2K(t-s)}\,\mathrm{d}s\right)\left(\int_0^t \|\boldsymbol{r}_s\|_2^2\,\mathrm{d}s\right). \tag{17}$$

The first factor is $C_K(t)^2$, including when $K = 0$. Taking expectations over $(\boldsymbol{x}, y) \sim \mathbb{P}$ and using $t \le 1$ bounds the expected value of the second factor by R_{FM}. Consequently,

$$\mathbb{E}_{\mathbb{P}}[\|\boldsymbol{e}_t\|_2] \le \left(\mathbb{E}_{\mathbb{P}}[\|\boldsymbol{e}_t\|_2^2]\right)^{1/2} \le C_K(t)\sqrt{R_{\mathrm{FM}}}. \tag{18}$$

It remains to relate path error to classification loss. For a fixed label y,

$$\nabla_{\boldsymbol{z}}\ell_{\mathrm{CE}}(\boldsymbol{z}, y) = \mathrm{softmax}(\boldsymbol{z}) - \boldsymbol{e}_y, \tag{19}$$

where $\boldsymbol{e}_y$ is the yth standard basis vector. The Euclidean distance between a probability vector and $\boldsymbol{e}_y$ is at most $\sqrt{2}$, so cross-entropy is $\sqrt{2}$-Lipschitz in the logits. It is also convex in the logits. Applying these two properties in turn yields

$$\begin{aligned}\ell_{\mathrm{CE}}(\boldsymbol{z}_t^{\boldsymbol{\phi}}, y) &\le \ell_{\mathrm{CE}}(\bar{\boldsymbol{z}}_t, y) + \sqrt{2}\|\boldsymbol{e}_t\|_2 \\ &\le (1-t)\ell_{\mathrm{CE}}(\boldsymbol{z}_0, y) + t\ell_{\mathrm{CE}}(\boldsymbol{z}_1, y) + \sqrt{2}\|\boldsymbol{e}_t\|_2.\end{aligned} \tag{20}$$

Taking expectations and substituting equation 18 proves the risk bound. The final claim follows by replacing both endpoint risks with ϵ. □

A.2 Closed-form affine flow matching

Consider the time-independent affine field $\boldsymbol{v}_{\boldsymbol{\phi}}(\boldsymbol{z}, t) = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$. Here $d = d_{\mathrm{out}}$ for logit-space flow matching. Arrange $n \ge 1$ paired endpoints as columns of $\boldsymbol{Z}_0, \boldsymbol{Z}_1 \in \mathbb{R}^{d\times n}$. These pairs may come from any fixed coupling. For sampled times $t_1, \dots, t_n \in [0, 1]$, define

$$\boldsymbol{T} = \mathrm{diag}(t_1, \dots, t_n), \qquad \boldsymbol{Q} = \boldsymbol{Z}_0(\boldsymbol{I} - \boldsymbol{T}) + \boldsymbol{Z}_1\boldsymbol{T}, \qquad \boldsymbol{D} = \boldsymbol{Z}_1 - \boldsymbol{Z}_0. \tag{21}$$

Thus the columns of $\boldsymbol{Q}$ are intermediate logits, and those of $\boldsymbol{D}$ are their target velocities. In this subsection, $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector and $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ is the identity matrix.

We first detail the sampled-time solution in Theorem 2, then give its uniform-time extension. For sampled times, the empirical objective is

$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{n}\|\boldsymbol{W}\boldsymbol{Q} + \boldsymbol{b}\mathbf{1}^\top - \boldsymbol{D}\|_F^2. \tag{22}$$

Define the column means and centered matrices by

$$\begin{aligned}\bar{\boldsymbol{q}} &= \tfrac{1}{n}\boldsymbol{Q}\mathbf{1}, & \bar{\boldsymbol{d}} &= \tfrac{1}{n}\boldsymbol{D}\mathbf{1}, \\ \boldsymbol{Q}_{\mathrm{c}} &= \boldsymbol{Q} - \bar{\boldsymbol{q}}\mathbf{1}^\top, & \boldsymbol{D}_{\mathrm{c}} &= \boldsymbol{D} - \bar{\boldsymbol{d}}\mathbf{1}^\top.\end{aligned} \tag{23}$$

Among all minimizers, the solution with minimum Frobenius norm of $\boldsymbol{W}$ is

$$\begin{aligned}\boldsymbol{W}^* &= \boldsymbol{D}_{\mathrm{c}}\boldsymbol{Q}_{\mathrm{c}}^\top(\boldsymbol{Q}_{\mathrm{c}}\boldsymbol{Q}_{\mathrm{c}}^\top)^\dagger, \\ \boldsymbol{b}^* &= \bar{\boldsymbol{d}} - \boldsymbol{W}^*\bar{\boldsymbol{q}}.\end{aligned} \tag{24}$$

Alternatively, average exactly over $t \sim \mathcal{U}[0, 1]$ for each fixed endpoint pair. With midpoint $\boldsymbol{m}_i = (\boldsymbol{z}_{0,i} + \boldsymbol{z}_{1,i})/2$ and displacement $\boldsymbol{d}_i = \boldsymbol{z}_{1,i} - \boldsymbol{z}_{0,i}$, this objective is

$$\bar{\mathcal{L}}_{\mathrm{FM}}(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{n}\sum_{i=1}^n \int_0^1 \left\|\boldsymbol{W}\left(\boldsymbol{m}_i + (t - \tfrac{1}{2})\boldsymbol{d}_i\right) + \boldsymbol{b} - \boldsymbol{d}_i\right\|_2^2 \mathrm{d}t. \tag{25}$$

Let

$$\boldsymbol{M} = \tfrac{1}{2}(\boldsymbol{Z}_0 + \boldsymbol{Z}_1), \qquad \bar{\boldsymbol{m}} = \tfrac{1}{n}\boldsymbol{M}\mathbf{1}, \qquad \boldsymbol{M}_{\mathrm{c}} = \boldsymbol{M} - \bar{\boldsymbol{m}}\mathbf{1}^\top, \tag{26}$$

and define

$$\begin{aligned}\boldsymbol{C}_{\mathrm{M}} &= \tfrac{1}{n}\boldsymbol{M}_{\mathrm{c}}\boldsymbol{M}_{\mathrm{c}}^\top, & \boldsymbol{C}_{\mathrm{D}} &= \tfrac{1}{n}\boldsymbol{D}_{\mathrm{c}}\boldsymbol{D}_{\mathrm{c}}^\top, \\ \boldsymbol{C}_{\mathrm{DM}} &= \tfrac{1}{n}\boldsymbol{D}_{\mathrm{c}}\boldsymbol{M}_{\mathrm{c}}^\top, & \boldsymbol{\Sigma}_{\mathrm{Q}} &= \boldsymbol{C}_{\mathrm{M}} + \tfrac{1}{12}(\boldsymbol{C}_{\mathrm{D}} + \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^\top).\end{aligned} \tag{27}$$

The minimizer with minimum Frobenius norm of $\boldsymbol{W}$ is then

$$\begin{aligned} \boldsymbol{W}^*_{\text{unif}} &= \boldsymbol{C}_{\text{DM}} \boldsymbol{\Sigma}^{\dagger}_{\text{Q}}, \\ \boldsymbol{b}^*_{\text{unif}} &= \bar{\boldsymbol{d}} - \boldsymbol{W}^*_{\text{unif}} \bar{\boldsymbol{m}}. \end{aligned} \tag{28}$$

Each objective has a unique minimizer when its input covariance is full rank. Otherwise, the pseudoinverse selects the minimizer with smallest $\|\boldsymbol{W}\|_F$; it need not minimize a joint norm of $\boldsymbol{W}$ and $\boldsymbol{b}$.

Proof of Theorem 2 and its uniform-time extension. Both formulas follow from the same least-squares argument: fit the intercept by matching means, then regress the centered target velocities on the centered inputs.

For sampled times, fix $\boldsymbol{W}$ and differentiate with respect to $\boldsymbol{b}$. Setting the derivative to zero gives

$$\boldsymbol{W}\bar{\boldsymbol{q}} + \boldsymbol{b} - \bar{\boldsymbol{d}} = \boldsymbol{0}, \tag{29}$$

so the unique optimal intercept for this $\boldsymbol{W}$ is $\boldsymbol{b} = \bar{\boldsymbol{d}} - \boldsymbol{W}\bar{\boldsymbol{q}}$. Substituting it into the objective leaves

$$\min_{\boldsymbol{W}} \frac{1}{n} \|\boldsymbol{W}\boldsymbol{Q}_{\text{c}} - \boldsymbol{D}_{\text{c}}\|_F^2. \tag{30}$$

The normal equations are

$$\boldsymbol{W}(\boldsymbol{Q}_{\text{c}}\boldsymbol{Q}_{\text{c}}^{\top}) = \boldsymbol{D}_{\text{c}}\boldsymbol{Q}_{\text{c}}^{\top}. \tag{31}$$

If $\boldsymbol{Q}_{\text{c}}\boldsymbol{Q}_{\text{c}}^{\top}$ is invertible, these equations determine $\boldsymbol{W}$ uniquely. If it is singular, the objective leaves the action of $\boldsymbol{W}$ on the null space unconstrained. The Moore–Penrose solution sets that action to zero and hence has minimum Frobenius norm. This gives the stated $\boldsymbol{W}^*$ and, by matching means, $\boldsymbol{b}^*$.

For the uniform-time objective, choose an endpoint pair uniformly from the n pairs and sample $t \sim \mathcal{U}[0,1]$ independently. Write its interpolated input as

$$\boldsymbol{q}_{i,t} = \boldsymbol{m}_i + s\boldsymbol{d}_i, \qquad s = t - \tfrac{1}{2}. \tag{32}$$

Let $\mathbb{E}_{i,t}$ denote averaging over both the pair index and time, and $\mathbb{E}_i$ averaging over the pairs alone. Since $\mathbb{E}_t[s] = 0$ and $\mathbb{E}_t[s^2] = 1/12$, the input mean is $\bar{\boldsymbol{m}}$, while the target mean is $\bar{\boldsymbol{d}}$. Matching these means gives $\boldsymbol{b} = \bar{\boldsymbol{d}} - \boldsymbol{W}\bar{\boldsymbol{m}}$.

To obtain the centered normal equations, we need the input covariance and the target–input cross-covariance. Expanding the centered input gives

$$\begin{aligned} \mathbb{E}_{i,t}\big[(\boldsymbol{q}_{i,t} - \bar{\boldsymbol{m}})(\boldsymbol{q}_{i,t} - \bar{\boldsymbol{m}})^{\top}\big] &= \boldsymbol{C}_{\text{M}} + \tfrac{1}{12}\mathbb{E}_i[\boldsymbol{d}_i\boldsymbol{d}_i^{\top}] \\ &= \boldsymbol{C}_{\text{M}} + \tfrac{1}{12}(\boldsymbol{C}_{\text{D}} + \bar{\boldsymbol{d}}\bar{\boldsymbol{d}}^{\top}) \\ &= \boldsymbol{\Sigma}_{\text{Q}}. \end{aligned} \tag{33}$$

The mixed terms vanish because s has zero mean and is independent of the pair index. For the same reason,

$$\mathbb{E}_{i,t}\big[(\boldsymbol{d}_i - \bar{\boldsymbol{d}})(\boldsymbol{q}_{i,t} - \bar{\boldsymbol{m}})^{\top}\big] = \boldsymbol{C}_{\text{DM}}. \tag{34}$$

The normal equations are therefore $\boldsymbol{W}\boldsymbol{\Sigma}_{\text{Q}} = \boldsymbol{C}_{\text{DM}}$. As in the sampled-time case, the pseudoinverse yields the solution with minimum Frobenius norm, and matching means recovers its intercept. Full rank gives uniqueness in either case. □

B Reproducibility

Environment All experiments were implemented in Python 3.13, with Jax v0.11.1 [CUDA driver version 13].

Datasets and preprocessing We used the standard MNIST and CIFAR-10 train/test splits. Images were converted to tensors and normalized using the standard dataset-specific mean and standard deviation. For CIFAR-10, training augmentation consisted of random 32×32 crops with four-pixel padding and random horizontal flips. No augmentation was used during evaluation.

Table 4: Trajectory straightness across flow-matching methods on ResNet-20, as given by equation 35. Lower values indicate straighter trajectories. Next to each value, we highlight the increase compared to OT-CFM. All methods are evaluated for merging ResNet-20 models trained on CIFAR-10.

Method	**Straightness Error** $\downarrow$
Independent FM	6.33×10^{-1} $(\times 81.7)$
Paired FM	1.04×10^{-2} $(\times 1.35)$
OT-CFM	7.75×10^{-3} $(\times 1)$

Architectures Unless otherwise specified, MLPs used hidden layers of width 512. We used ResNet-20 as is, without any changes to the architecture.

Training Models were trained with Adam using batch size 512 and learning rate $\eta = 10^{-3}$. The learning rate was decayed using a cosine schedule.

Label-skewed splits For complementary-data experiments, the first model was trained using 80% of examples from labels 0–4 and 20% from labels 5–9. The second model used the complementary subset. The partition was generated deterministically from a fixed seed, to ensure the training samples were shared across runs.

Git Re-Basin Git Re-Basin permutations were optimized with the Hungarian algorithm for up to 100 iterations.

Flow matching head Unless otherwise stated, the flow matching head is a single affine layer. We use SiLU for modeling the nonlinearities wherever applicable, the hidden dimension being equal to the output dimension. The model was trained with Adam initialized with a learning rate $\eta = 10^{-3}$ and was trained for 10 training epochs. Learned flows were evaluated using Euler integration with 50 steps.

C STRAIGHTNESS OF TRAJECTORIES

Let $\mathbf{z}_t^{\phi}$ denote the learned trajectory when its initial logits are sampled from the source distribution. We measure its deviation from constant-velocity motion by

$$S = \int_0^1 \mathbb{E}\left[\left\|(\mathbf{z}_1^{\phi} - \mathbf{z}_0^{\phi}) - \dot{\mathbf{z}}_t^{\phi}\right\|_2^2\right] \mathrm{d}t. \tag{35}$$

The expectation is over source samples, and both endpoints belong to the learned trajectory. Lower values indicate straighter trajectories.

Alongside optimal-transport coupling, we evaluate two ways to pair training endpoints:

- **Independent coupling**: Sample $\boldsymbol{z}_0 \sim p_0$ and $\boldsymbol{z}_1 \sim p_1$ independently.
- **Same-input coupling**: For each calibration input $\boldsymbol{x}$, pair the logits $\boldsymbol{z}_0(\boldsymbol{x})$ and $\boldsymbol{z}_1(\boldsymbol{x})$. We refer to this variant as Paired FM in the table. This is the default pairing that we use.
- **Optimal Transport**: Inspired by Tong et al., for each minibatch $(\boldsymbol{z}_0(\boldsymbol{x}), \boldsymbol{z}_1(\boldsymbol{x}))$, we output the

We find that OT-CFM produces straight trajectories without sacrificing accuracy in these experiments. This implies that it would be extremely beneficial for