When is Model Souping Tasty? Similarity, Transitivity, and Robustness

Pierre Mackenzie
Department of Computer Science,
University of British Columbia
Vancouver, Canada
pierrerl@cs.ubc.ca

Simon Ghyselincks
Department of Computer Science,
University of British Columbia
Vancouver, Canada
Vector Institute
Toronto, Canada

Evan Shelhamer
Department of Computer Science,
University of British Columbia
Vancouver, Canada
Vector Institute
Toronto, Canada

Abstract

Model souping is a post-training technique where the parameters of models are averaged, often leading to improved performance over constituent models without increasing inference cost. However, the specific conditions required for success are not well understood, particularly regarding the trade-off between model diversity and stability. We analyse over 5,000 two-model ResNet-50 soups trained on CIFAR-100, with diversity controlled by branching ingredients from a shared training trajectory at varying epochs. We find that effective souping requires a balance: models must be similar enough to avoid model collapse, but diverse enough to yield improvements. We show that we can predict soupability relatively well with standard similarity metrics. Furthermore, we provide empirical evidence for the hypothesis that souping works by averaging within a low-loss basin by showing that souping is moderately transitive. We also observe that soup gains on corrupted data are strongly correlated with those on in-distribution data. Our findings yield practical advice for machine learning practitioners: if you want a tasty soup, use the right cooking time! Code and experiments are available at: https://github.com/chipnbits/too-salty.

Keywords: Machine Learning, Model Souping

1 Introduction

Wortsman et al. [11] introduce *souping*: averaging the model parameters produced by different fine-tuning trajectories from a common pre-trained model can yield better generalization than any one ingredient. Souping is formulated as a linear interpolation of model *parameters* θ using *weights* α. For the case of two models, $\theta_{\text{soup}} = (1-\alpha)\theta_A + \alpha\theta_B$, creating a new 'souped' model. This not only captures the benefit of multiple optimization paths but also introduces an additional adaptation opportunity, as Croce et al. [2] suggest that dynamically adjusting the interpolation weights enables intermediate behaviours that can better match a range of distributional shifts. In contrast to ensembling which combines the *outputs* of models, souping combines the model *parameters.* This has the benefit of maintaining the same computational cost of inference as a single model.

Wortsman et al. [11] hypothesise that souping works because fine-tuned models often lie within the same low-loss basin. A convex combination of their parameters is expected to remain within this basin while reducing the variance introduced by noisy training. They find that when the angle formed by the pre-trained model and ingredients to be souped together is larger, implying greater diversity, there is a greater performance boost from souping. Understanding why averaging maintains a low loss, how diversity contributes to robustness, and when souping is beneficial is therefore essential for studying adaptation more broadly.

1.1 Related Work

Souping has been used in a variety of settings. Croce et al. [2] soup models trained to be robust to different distribution shifts and Ramé et al. [10] soup ingredients trained on different tasks. Both cases lead to better generalisation. Jang et al. [9] show that fine-tuned parameters are distributed around a high-accuracy center and use the angle between as few as two fine-tuned weights and their shared branch point to approximate this low-loss center.

Souping is related to the idea of SWA (Stochastic Weight Averaging) [8]. In SWA, the ingredients of the soup come from different steps along the same training trajectory. By contrast, souping averages models from independent training trajectories from a shared initialization.

Souping success requires models to be 'compatible' in the sense that averaging their weights has low loss. This notion is closely related to 'stability to SGD (Stochastic Gradient Descent) noise' Frankle et al. [4], enabled by linear mode connectivity between models trained from the same initialisation under different SGD noise. Such stability is only present after sufficient shared training Iyer et al. [7]. We propose that this defines a window during training in which souping is effective and that pre-emptive souping leads to a high loss barrier, or 'model collapse'.

1.2 Our Contributions

Following these works, we seek to better understand souping by conducting a series of experiments addressing the following questions:

1. How much shared training is required for souping to be effective? We investigate varying the number of shared pre-training epochs before splitting into fine-tuned variants to empirically map the transition from model collapse to effective souping.

2. Can model similarity predict the effectiveness of souping? Identical models yield no generalization benefit, while overly dissimilar models fail to inhabit a shared low-loss region. We investigate whether standard similarity metrics can predict this balance between model compatibility and diversity.

3. Is souping transitive? If model A soups with B, and B soups with C, will A soup with C?

4. Does souping in-distribution predict souping out-of-distribution? While souping has been shown to help with robustness to distribution shifts, we seek to answer how correlated the soup gains are between in-distribution and out-of-distribution data.

5. How does souping compare to other neural averaging techniques? We empirically compare our binary soups to SWA.

More experiments on the effect of permuting models [1] prior to souping and how souping affects robustness can be found in Appendix A.1 and Figure A14 respectively.

1.3 Soups and Soup Gain

We soup pairs of models using the simple arithmetic mean $\theta_{\text{soup}} = \frac{1}{2}\theta_A + \frac{1}{2}\theta_B$. As shown by Ainsworth et al. [1] and [7], the loss barrier typically has the most extreme behaviour at the midpoint, making it a useful summary statistic of souping performance across weights interpolation, accurately identifying souping failure. Exemplary weighting plots are shown in A15 to further support this assumption.

We define *Soup gain* as the test set accuracy gain of souping two models relative to the *best* parent model.

$$\text{soup gain} = \text{acc}(\theta_{\text{soup}}) - \max\{\text{acc}(\theta_A), \text{acc}(\theta_B)\}$$

where acc(.) denotes test accuracy. We compare against the maximum accuracy of the ingredients, rather than the mean, as the purpose of a soup should be to improve over its ingredients. Soup gain can alternatively be defined as the decrease in loss relative to the best ingredient. We use soup gain as the primary measure of the effectiveness of souping. We also refer to soups with positive and negative soup gain simply as positive and negative soups.

2 Experiments

We train a baseline model for image classification on the CIFAR-100 dataset with ResNet-50 [5] following [3], saving checkpoints every 10 epochs. From each checkpoint we train 4 new models with different optimizer settings, for details see Appendix Tables 2 and 3. All models are trained to convergence, with the best validation scored model saved for experiments, as shown in Figure 1. A total of 4 variants and 26 branch points were trained, yielding 104 related models and 5, 356 binary souping combinations for analysis. Additionally, we train 12 baseline models with and without Stochastic Weight Averaging (SWA) to compare the depth of fine-tuning paths against the breadth of SWA, see Appendix A.3.

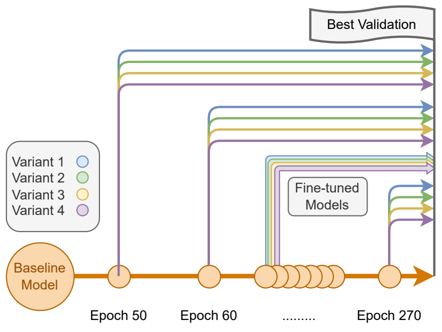

Figure 1. Branching of fine-tuned models from baseline checkpoints. A single baseline model is trained, with checkpoints saved every 10 epochs. From each checkpoint, 4 variants are trained with different optimizer hyper-parameter perturbations.

2.1 Soup Gain and Shared Epochs

Figure 2 shows the cumulative distribution function (CDF) of soup gains. Many soups have extreme behaviour, with 40% of soups losing over 60% and only 14% with positive gain. There is also a sharp transition to collapse, with only 15% soups between −70% and −20% gain. Figure 3 groups soups by the number of shared training epochs between ingredients (e.g., ingredients branched at epochs 50 and 100 share 50 epochs). We observe that the probability of positive soup gain generally increases with shared epochs, reaching around 80% after 260 shared epochs. At this stage, almost no soups drop accuracy by more than 5%. However, peak gains (> 0.5%) require a balance; if shared training is too extensive ingredients are not diverse enough, performance diminishes. Further analysis of the distribution of soups gains over shared epochs on mean gain, loss metrics, and model collapse is provided in Appendices A3, A2 and A4.

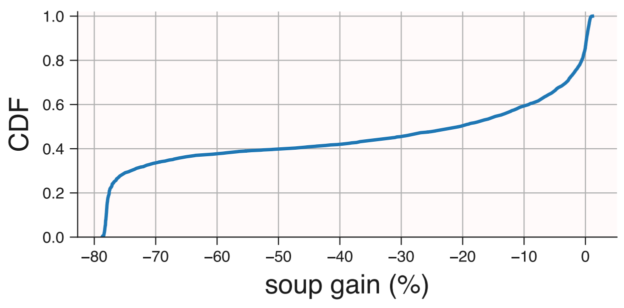

Figure 2. Empirical CDF of soup gain over all soups.

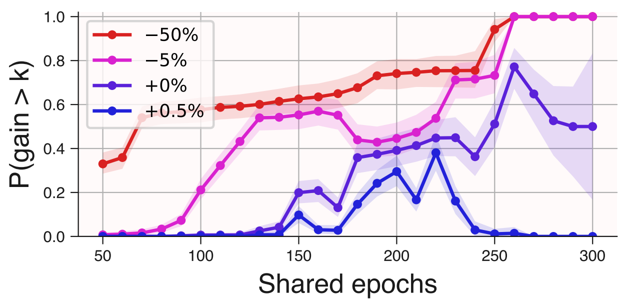

Figure 3. Probability of soup gain being greater than k% for varying k with 95% CIs.

2.2 Predicting Soupability with Similarity

To test if soupability is predictable, we compute a variety of similarity and distance metrics between model pairs. All metrics perform similarly, as seen in Figure A8. We plot the KL divergence between the outputs of the ingredients as a representative example in Figure 4 and find a strong negative correlation with soup gain (Spearman −0.86), indicating divergence often leads to model collapse. However, among positive soups, soup gain has a moderate positive correlation with dissimilarity (Spearman 0.39), see Figure 5. Effective souping requires models to be sufficiently similar, but also rewards variety. Balancing these two effects is key to tasty soups. Additionally, shared epochs correlate strongly with all similarity metrics, for example (Spearman −0.67) with KL divergence. More details can be found in Appendix A5.

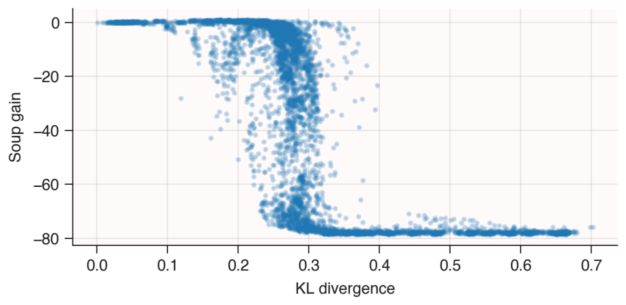

Figure 4. KL vs soup gain (Spearman −0.86).

2.3 Is Souping Transitive?

To test the hypothesis that soupable models reside in the same low-loss basin, we evaluate the transitivity of model triplets (A, B, C). Figure 6 shows that B and C are highly likely to soup only if A soups with both B and C. We observe a moderate positive correlation (Spearman 0.64) between the gain of (B, C) and the minimum gain of (A, B) and (A, C), see Figure A10. We attempt to explain when transitivity fails in terms of the branching epochs of the vertices of a given triple, but find no clear pattern. See Appendix Tables 4 and 5.

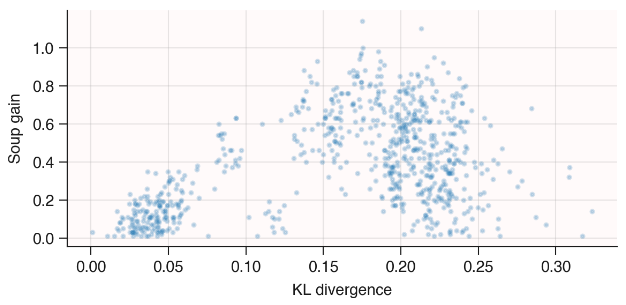

Figure 5. KL vs (+) soup gain (Spearman 0.39).

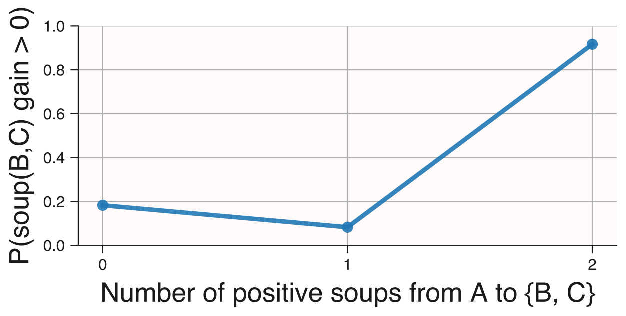

Figure 6. Probability of positive soup gain of B and C vs positive soups with A.

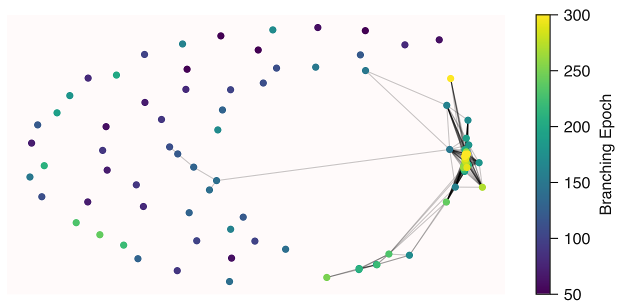

Figure 7. 2D embedding of 104 models using soup-gain distance; edges indicate (+) soups.

We embed our 104 models into 2D using soup gain as a distance metric to search for souping clusters, plotting them in Figure 7, with details found in Section A.5. We find all most successful soups form the edges of a single connected component of models, while there is a single dense cluster of models which are largely branched from later on in the training procedure. There are many counter-examples, but this supports the conclusion that souping is moderately transitive and successful ingredients generally lie within a single loss basin.

Type	P(Gain > 0)	Mean Gain (Acc %)	Mean Corrupted Gain (Acc %)
Soups: 200	43.6% ± 7.7%	0.6% ± 0.04%	0.71% ± 0.09%
Soups: 250	51.2% ± 9.9%	0.16% ± 0.04%	0.05% ± 0.08%
SWA (12)	100%	1.6% ± 0.3%	3.7% ± 0.4%

Table 1. Comparison of souping robustness to SWA.

2.4 Souping for Robustness to Corruption

To establish whether souping for in-distribution (ID) performance also increases out-of-distribution (OOD) performance, we compute the soup gain on CIFAR-100C [6] with severity level 3. The soup gains on test and corrupted data correlate strongly (Spearman 0.99), even when restricted to positive soups (Pearson 0.61), see Figures A11, A13. Thus, ID performance improvement can transfer to unseen target distributions.

We also plot the probability of positive soup gain on corrupted data as a function of shared epochs in Figure A12. The probability of positive gain increases with the number of shared epochs for both clean and corrupted data but the corrupted data always has a lower probability of positive gain.

2.5 How does souping compare to SWA?

Model accuracy on both clean and corrupted data has been shown to improve with souping. Here we compare the improvements of souping to SWA in Tab. 1. SWA offers a significant boost in robustness to shift, always improves the baseline run, and performs better than our best geometric mean soups. We hypothesize that in this experimental setting, SWA has explored a wider breadth of points in the low loss basin than possible with just two ingredients and has more closely converged to the low-loss center described by Jang et al. [9]. More carefully selecting ingredients for a soup, either through greedy souping with more models [11] or by using a more sophisticated weighting scheme [2] may reduce the gap.

3 Conclusion

In this work, we have conducted a series of experiments to further our understanding of souping. We find that souping is more effective when models share more training epochs before diverging into different training trajectories. This suggests that shared training is necessary to reach a common low-loss basin. However, too many shared epochs and the soup gain is minimal. We also find that various similarity measures between models correlate similarly with soup gain. More similar models are less likely to collapse when souped, but also yield smaller soup gains. Thus, the right balance must be struck for the most effective souping. When souping, we encourage practitioners to test a range of similarities of ingredients to ensure they are finding an optimal soup. Our experiments showing that souping is mostly transitive support the low-loss basin hypothesis. Finally, we find that soup gains on in-distribution data are strongly correlated with those on corrupted data.

Limitations: While our experiments provide insight into souping, they are limited in scope. We only consider one dataset (CIFAR-100), one architecture (ResNet-50) with one baseline training trajectory. It is possible that our findings do not generalise to other datasets or architectures. Additionally, we only consider pairwise souping using the arithmetic mean at the midpoint. Other methods of souping, such as learned soups, may yield different results.

Future Work: Future empirical work could conduct similar experiments in different settings, such as a variety of model architectures and datasets. It could also center on more thoroughly analysing gains of souping relative to SWA. Theory could be developed for souping in simpler settings like an overparameterized linear model or a shallow network. Theory could also be created to help characterise the noise reduction and consequently the reduction in loss we expect from souping.

References

[1] Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2023. Git Re-Basin: Merging Models modulo Permutation Symmetries. arXiv:2209.04836 [cs.LG] https://arxiv.org/abs/2209.04836

[2] Francesco Croce, Sylvestre-Alvise Rebuffi, Evan Shelhamer, and Sven Gowal. 2023. Seasoning Model Soups for Robustness to Adversarial and Natural Distribution Shifts. arXiv:2302.10164 [cs.LG] https://arxiv.org/abs/2302.10164

[3] Eduardo Dadalto. 2023. ResNet-50 Model Trained on CIFAR-100. https://huggingface.co/edadaltocg/resnet50_cifar100. Hugging Face Model Repository.

[4] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2020. Linear Mode Connectivity and the Lottery Ticket Hypothesis. arXiv:1912.05671 [cs.LG] https://arxiv.org/abs/1912.05671

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV, USA) *(CVPR '16)*. IEEE, Las Vegas, NV, USA, 770–778. doi:10.1109/CVPR.2016.90

[6] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. arXiv:1903.12261 [cs.LG] https://arxiv.org/abs/1903.12261

[7] Gaurav Iyer, Gintare Karolina Dziugaite, and David Rolnick. 2024. Linear Weight Interpolation Leads to Transient Performance Gains. *Transactions on Machine Learning Research* (2024). https://openreview.net/forum?id=XGAdBXlFcj Presented at HiLD (ICML 2024 Workshop).

[8] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2019. Averaging Weights Leads to Wider Optima and Better Generalization. arXiv:1803.05407 [cs.LG] https://arxiv.org/abs/1803.05407

[9] Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. 2025. Model Stock: All we need is just a few fine-tuned models. arXiv:2403.19522 [cs.LG] https://arxiv.org/abs/2403.19522

[10] Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. 2023. Model Ratatouille:

Recycling Diverse Models for Out-of-Distribution Generalization. arXiv:2212.10445 [cs.LG] https://arxiv.org/abs/2212.10445

[11] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. arXiv:2203.05482 [cs.LG] https://arxiv.org/abs/2203.05482

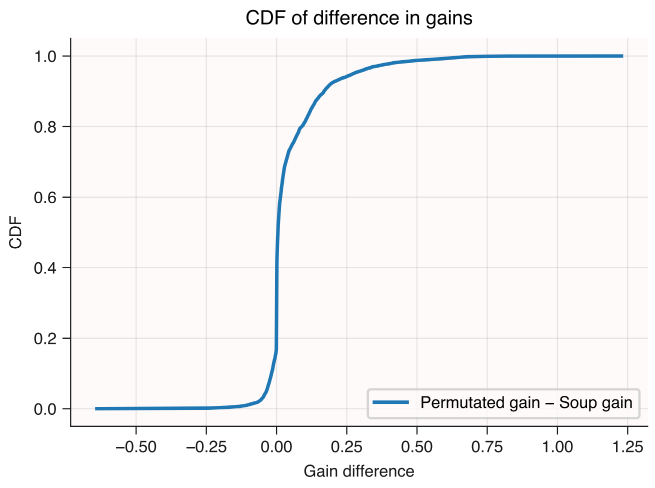

Figure A1. Cumulative distribution function (CDF) of the difference in soup gain before and after permutation alignment using `rebasin`. This ignores the 7% of soups with a loss higher than 5 after permutation as these make the plot difficult to interpret. The remaining mean and median difference is approximately zero, indicating that permutation alignment does not have a significant effect on the effectiveness of souping in our experiments. While some soups benefit from permutation alignment, others are negatively affected, leading to an overall negligible impact while there is a risk of severe degradation.

Model	Learning Rate Scale	Momentum Scale	Weight Decay Scale
Model 1	0.7073	1.1009	0.8284
Model 2	1.2244	0.9247	1.1150
Model 3	0.5112	1.0695	1.1099
Model 4	0.5373	0.8594	0.9078

Table 2. Optimizer perturbation scales applied during finetuning from the baseline ResNet-50 checkpoint on CIFAR-100. Each model scales the original SGD hyperparameters multiplicatively.

A Appendix

A.1 Permutation Alignment for Souping

Following the work from Ainsworth et al. [1], we investigate whether permuting the neurons of models prior to souping increases the effectiveness of souping. We use the `rebasin`[1] package to align pairs of models before souping. This package uses the 'matching weights' method which permutes the neurones by inspecting only the weights. This contrasts with 'activation matching' which requires forward passes through the network, and 'straight through estimators' which are even more computationally expensive. The authors find that matching weights performs similarly to activation matching while being computationally cheaper. Therefore, we only consider the matching weights method.

We align all 5, 346 pairs of models using `rebasin` and compute the soup gain after alignment. Prior to permutation, 14.25% of soups were positive, while after permutation, 14.32% of soups were positive. However, 7% of soups obtained a loss higher than 5, which is worse than the loss of any previous soup. We plot the cumulative distribution function (CDF) of the difference in soup gain before and after permutation in Figure A1. This plot shows that while permuting can sometimes help, it does not do so consistently. Further, the median difference in soup gain is approximately zero, indicating that permuting does not have a significant effect on the effectiveness of souping in our experiments. There also remains a significant risk of severe degradation. Thus, we conclude that 'matching weights' permutation does not make a noticeable different to soupability in our setting.

[1]https://pypi.org/project/rebasin/

Component	Hyperparameter	Value
Dataset	Dataset	CIFAR-100
	# Classes	100
	Data augmentation	Mirroring and Padded Offset
	Validation split	5% of training set
	Split seed	42
Model	Architecture	ResNet-50
	Pretrained	No (from scratch)
Optimization	Optimizer	SGD (Nesterov)
	Initial learning rate	0.1
	Momentum	0.9
	Weight decay	5×10^{-4}
Learning rate schedule	Scheduler	CosineAnnealingLR
	$T_{\max}$	280 epochs
	$\eta_{\min}$	0
Training	Epochs	300
	Batch size	128
	Mixed precision	No

Table 3. Training hyperparameters for CIFAR-100 with ResNet-50. A randomization seed of 42 is used unless otherwise specified.

A.2 Further Details on Experiments

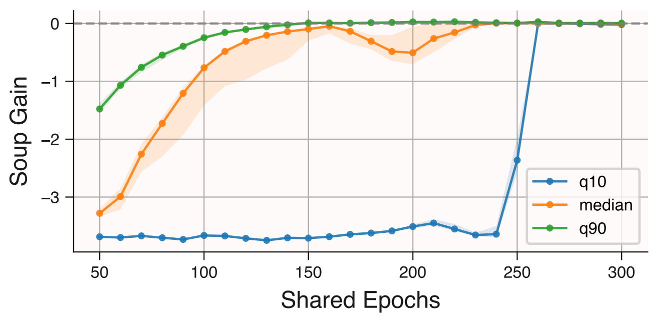

Figure A2. Quantiles of soup gain vs shared epochs with soup gain measured as the reduction in loss vs the best parent. Similar to the conclusions from 3, we find model collapse almost never occurs past shared epochs 250.

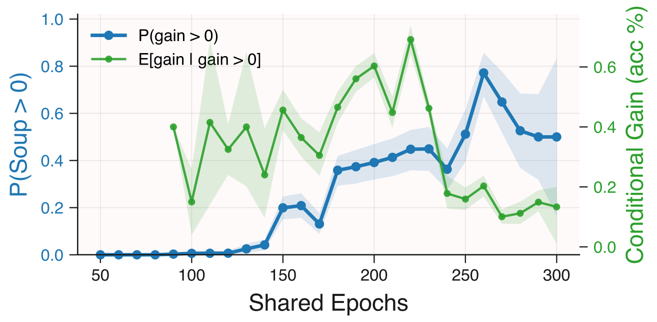

Figure A3. Probability of positive soup gain and conditional expected gain vs shared epochs. We find that the expected soup gain noisy, but is maximised in an ideal window of shared epochs. Past this point, and models are too similar, leading to minimal gains. Before this point, models are incompatible and rarely yield positive gain.

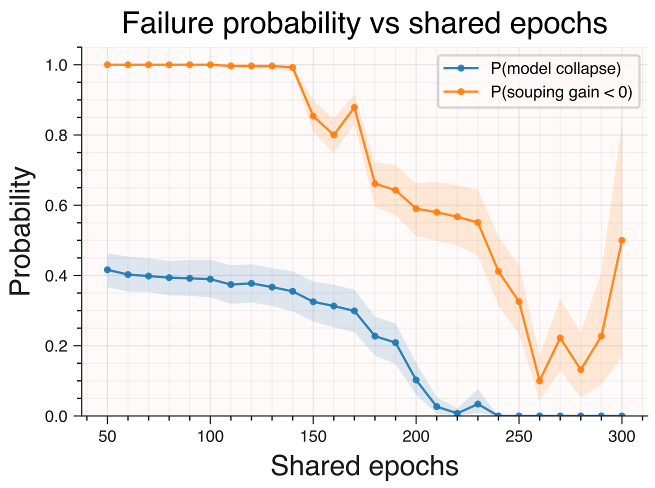

Figure A4. We plot both the probability of model collapse and the probability that soup gain is positive over varying shared epochs, with 95% bootstrapped confidence intervals. Model collapse is defined as the soup attaining less than 5% test accuracy. Both model collapse and negative soup gain decrease with shared epochs. In particular, model collapse is very rare past epoch 200 and never occurs past epoch 250.

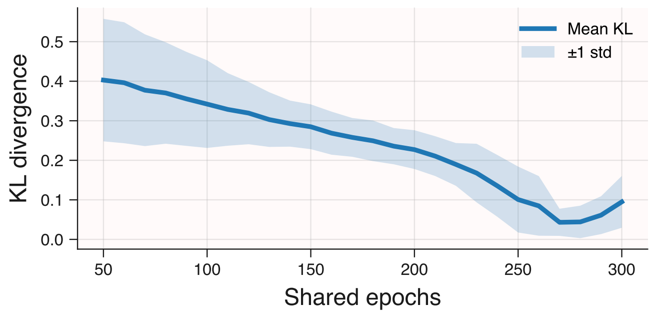

Figure A5. Shared epochs vs KL divergence. We plot the mean and standard deviation over each value of shared epochs. The full data has a Spearman correlation of −0.67. We conclude that we can noisily recover the number of shared epochs by measuring the KL divergence, and that model similarity can be controlled imprecisely by varying the number of shared epochs.

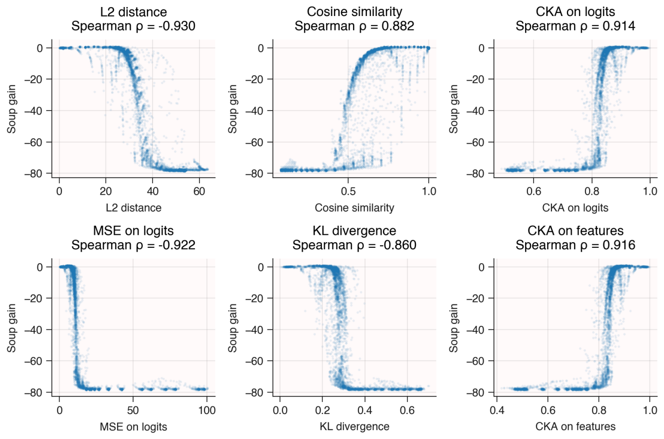

Figure A6. Soup gain vs various similarity and distance metrics between pairs of models. Each subplot shows the soup gain against one metric, with the Spearman correlation. All metrics perform similarly. We conclude that the more similar models are, the more likely souping is to not cause model collapse.

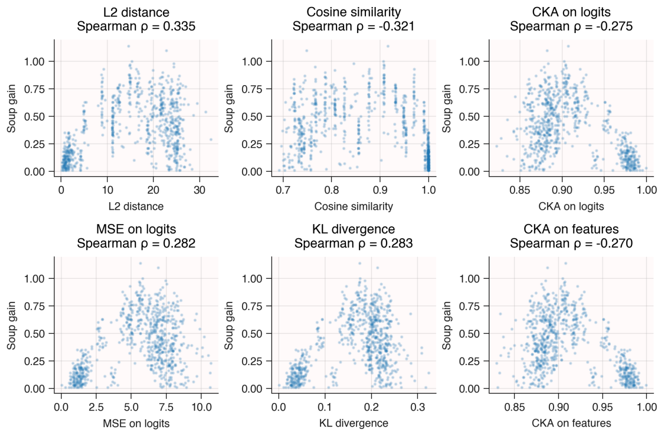

Figure A7. Soup gain vs various similarity and distance metrics between pairs of models, for the subset of soups with positive soup gain. Each subplot shows the soup gain against one metric, with the Spearman correlation. Each metric correlates similarly with soup gain. We see that when models are very similar, soup gain is small, while more dissimilar models can yield larger soup gains.

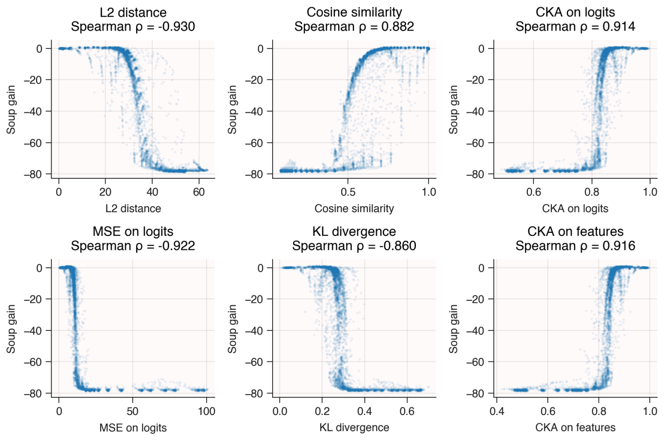

Figure A8. Soup gain vs various similarity and distance metrics between pairs of models. Each subplot shows the soup gain against one metric, with the Spearman correlation. All metrics perform similarly. We conclude that the more similar models are, the more likely souping is to not cause model collapse.

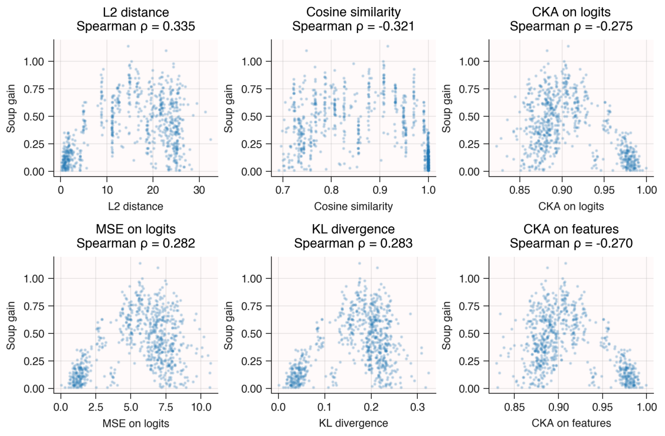

Figure A9. Soup gain vs various similarity and distance metrics between pairs of models, for the subset of soups with positive soup gain. Each subplot shows the soup gain against one metric, with the Spearman correlation. Each metric correlates similarly with soup gain. We see that when models are very similar, soup gain is small, while more dissimilar models can yield larger soup gains.

Table 4. Transitivity failure vs shared epochs over all (A, B, C) triples. Here n denotes the number of triples in the bin for which two soups are positive, and p_{fail} is the proportion of those triples for which the remaining soup is negative. Failure rates are lowest for very small and very large shared prefixes, with a peak at intermediate shared epochs, but no monotonic trend is observed.

Shared epochs (ABC)	n	p_{fail}
150–180	2294	0.071
180–200	1887	0.075
200–220	1657	0.119
220–240	1244	0.104
240–300	1369	0.055

Table 5. Transitivity failure vs branching-epoch difference $|B–C|$. Here n and p_{fail} are defined as in Table 4. We restrict to triples with shared epochs in $[200, 220]$, fix the lowest-epoch model, and vary the branching-epoch difference of the remaining two models. We observe no clear dependence of transitivity failure on epoch difference within this regime. Similar trends were observed for the other bins which are omitted for brevity.

Epoch diff (BC)	n	p_{fail}
0–10	792	0.124
10–20	497	0.123
20–30	419	0.124
30–50	578	0.080
50–100	369	0.100

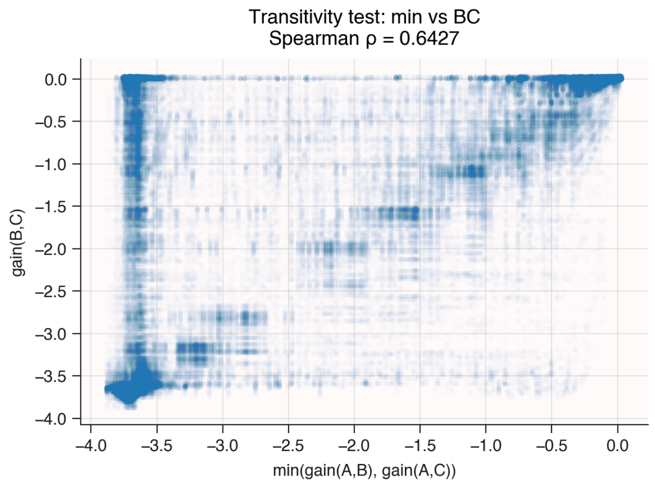

Figure A10. Scatterplot of the soup gain of models B and C against the minimum soup gain of models A with B and C. Each point represents a triplet of models (A, B, C). We observe a positive Spearman correlation of 0.64, suggesting that souping is fuzzily transitive. The correlation with the mean soup gain is lower, at 0.49.

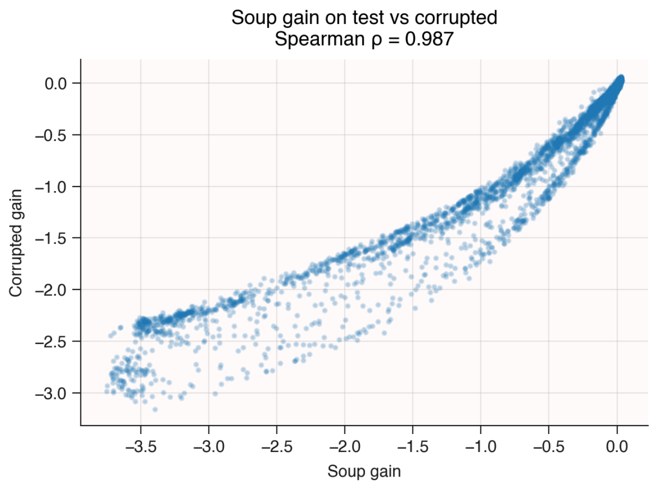

Figure A11. Scatterplot of the soup gain on test vs corrupted data. There is a clear sub-linear trend with strong correlation. Such a close relationship is sensitive to the nature of the distribution shift. This plot mostly shows that when model collapse occurs on the original test set, it also occurs on the corrupted data.

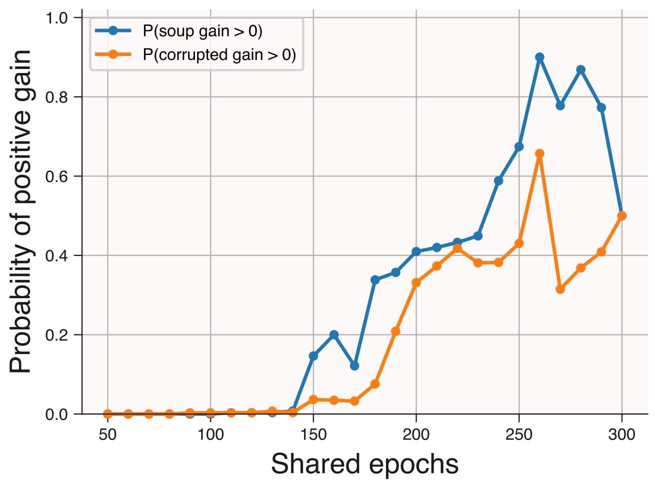

Figure A12. Probability of positive gain for soups as a function of the number of shared epochs. We see that the probability of positive gain increases with the number of shared epochs, for both clean and corrupted data. However,the corrupted data consistently has a slightly lower probability of positive gain. Thus, while souping also helps on corrupted data, it is slightly less effective than on clean data.

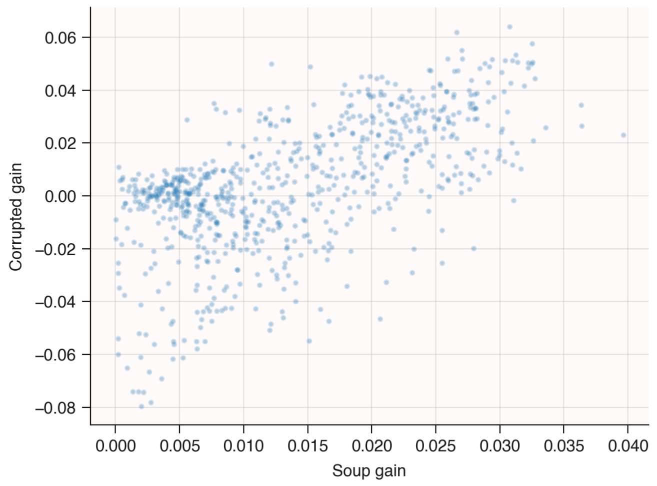

Figure A13. Plot of soup gain on test vs corrupted data for only models with positive soup gain on the test set. Spearman correlation of 0.61.

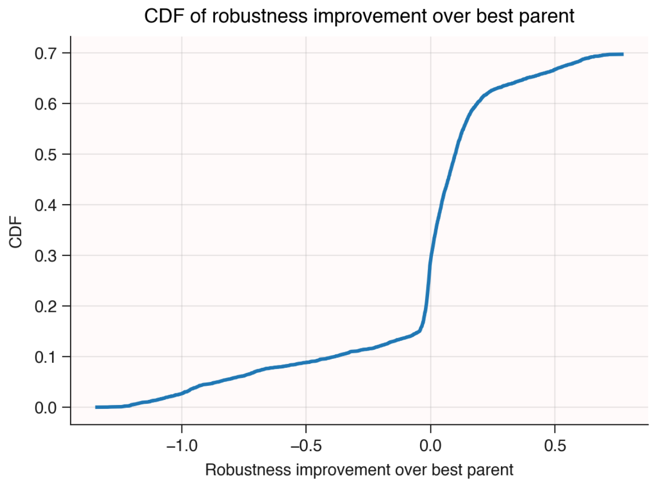

Figure A14. CDF of the difference in *robustness gap* before and after souping. The robustness gap is defined as the difference in loss between the test set and the corrupted set. The robustness gap before souping is taken as the minimum robustness gap of the parents. We see a fairly symmetric distribution. It has mean -0.04 and median 0.02. We therefore conclude that souping does not systematically improve the robustness gap.

A.3 SWA Experiment Runs

Baseline training was performed using the same hyper-parameters as in 3 and using the top validation model. We compare to SWA starting at epoch 220 from the same base model. The initialization and batch ordering seeds with full results are shown below in 6.

	Clean Data						Corrupted Data					
	Accuracy (↑)			Loss (↓)			Accuracy (↑)			Loss (↓)		
Seed	Base	SWA	Δ	Base	SWA	Δ	Base	SWA	Δ	Base	SWA	Δ
42	78.39	79.84	**+1.45**	1.02	0.90	**-0.12**	50.88	54.05	**+3.17**	2.37	2.63	+0.26
43	78.04	79.81	**+1.77**	1.03	0.87	**-0.16**	50.82	54.47	**+3.65**	2.35	2.51	+0.16
44	78.87	80.64	**+1.77**	1.00	0.86	**-0.14**	51.31	55.26	**+3.95**	2.36	2.57	+0.21
45	78.57	80.18	**+1.61**	1.00	0.86	**-0.14**	50.76	54.59	**+3.83**	2.39	2.60	+0.21
46	78.04	79.42	**+1.38**	1.02	0.89	**-0.13**	51.06	54.46	**+3.40**	2.35	2.57	+0.22
47	78.47	80.38	**+1.91**	1.01	0.88	**-0.13**	51.51	55.28	**+3.77**	2.35	2.56	+0.21
48	78.21	79.85	**+1.64**	1.02	0.90	**-0.12**	50.37	54.29	**+3.92**	2.40	2.62	+0.22
49	78.57	80.27	**+1.70**	1.01	0.88	**-0.13**	51.62	56.05	**+4.43**	2.33	2.44	+0.11
50	78.97	79.88	**+0.91**	1.00	0.89	**-0.11**	51.59	55.00	**+3.41**	2.34	2.56	+0.22
51	78.09	79.87	**+1.78**	1.03	0.89	**-0.14**	51.52	55.25	**+3.73**	2.32	2.52	+0.20
52	78.48	79.86	**+1.38**	1.02	0.89	**-0.13**	51.05	54.22	**+3.17**	2.37	2.65	+0.28
53	78.55	80.26	**+1.71**	1.02	0.88	**-0.14**	51.77	55.42	**+3.65**	2.31	2.50	+0.19
Mean	78.44	80.02	**+1.58**	1.01	0.88	**-0.13**	51.19	54.86	**+3.67**	2.35	2.56	+0.21
Std	0.30	0.33	0.27	0.01	0.01	0.01	0.43	0.60	0.36	0.03	0.06	0.04

Table 6. Comparison of basline and SWA models branching from Epoch 220.

A.4 Binary Model Soup Accuracy Curves

While our experiments make a statistical analysis on the characteristics of even-weighted binary soups, we verify the validity of our simplifying assumptions with characteristic accuracy curves across 16 binary pairs to support our claim that even-weighted soups are a good measure for performance. For soups exhibiting model collapse, the measure is much more accurate, while for positive gain soups the metric is indicative of soupability but not as accurate for actual souping gain.

A.5 Details for Model Embedding Creation

Given our evidence for transitivity, we consider the broader landscape of all 104 of our originally trained models. Do the soups all exist in separate clusters of models in separate loss basins? We define a distance metric defined as

$$d_{AB} = -\operatorname{sign}(\text{soup gain}) - 0.1 * \text{soup gain}$$

where d_{AB} is the distance between models A and B. Intuitively, this metric puts models close together that soup together positively, taking into account the magnitude of soup gain. We then cast this down into a 2-dimensional embedding using Multidimensional Scaling. We also color by branching epoch and mark edges that represent successful soups. The resulting plot is shown in Figure 7.

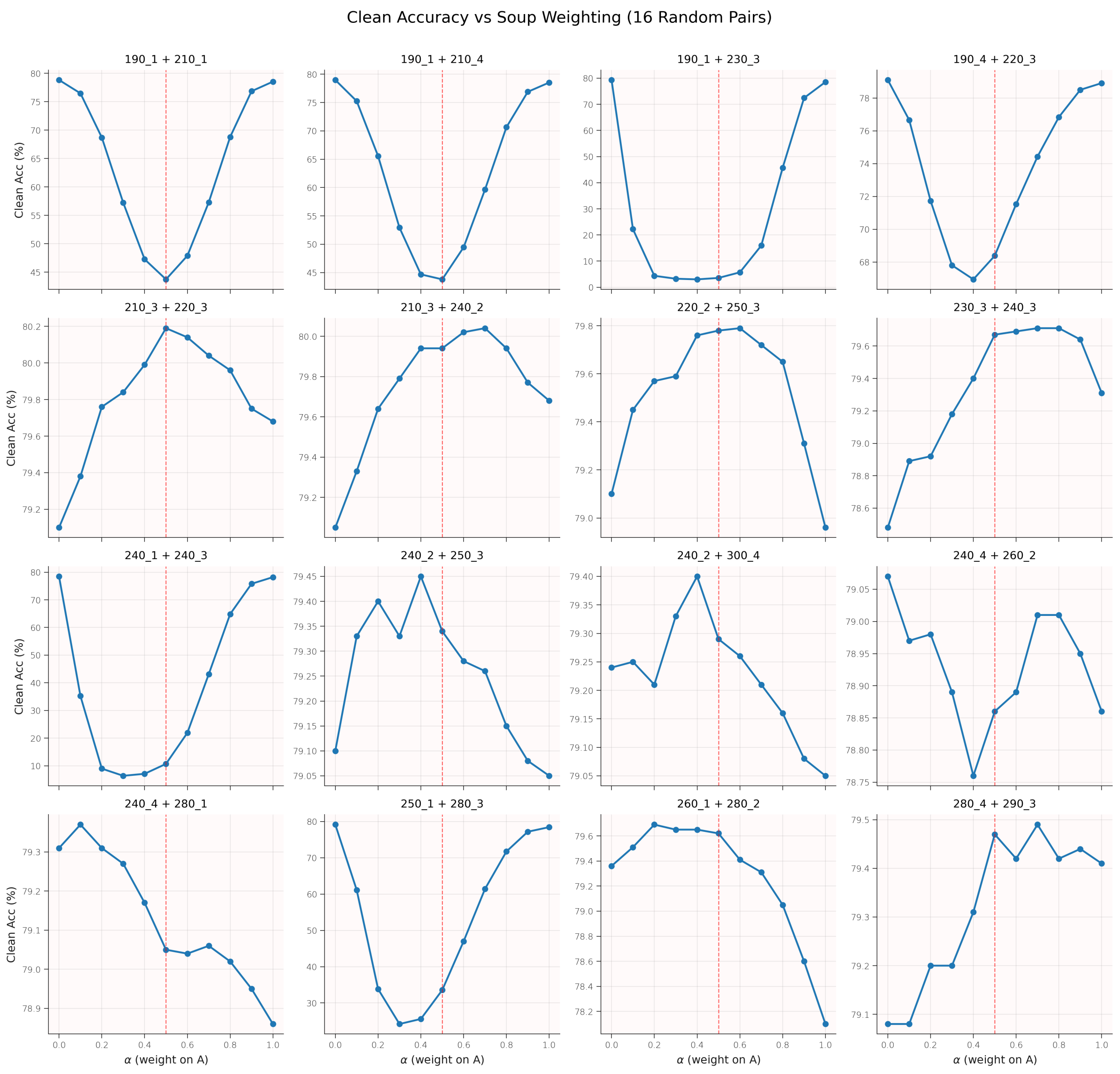

Figure A15. Souping characteristics across a range of parameter interpolation values for 16 models.