WHEN IS MODEL SOUPING TASTY? SIMILARITY, TRANSITIVITY, AND ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model souping is a technique in which 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 binary 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. Furthermore, we provide empirical evidence for the hypothesis that souping works by averaging within a low-loss basin. We also observe that soup gains on corrupted data are strongly correlated with those on in-distribution data. Finally, we compare souping to other parameter averaging methods and find that XXX. Code and experiments are available at: `https://anonymous.4open.science/r/too-salty-478E/`.

1 INTRODUCTION

Wortsman et al. (2022) introduced the idea of *souping* by showing that averaging model parameters produced by different fine-tuning trajectories often leads to better generalization than any individual ingredient. This not only captures the benefit of multiple adaptation paths but also introduces an additional adaptive mechanism, as Croce et al. (2023) suggest that dynamically adjusting soup weights enables intermediate behaviours that can better match a range of distributional shifts. In contrast to ensembling which combines the *outputs* of models, souping does not increase the computational cost of inference, requiring only a single forward pass.

Wortsman et al. (2022) hypothesise that souping works because fine-tuned models often lie in the same low-loss basin. The convex combination of their weights is expected to remain in the low-loss basin while reducing variance introduced by a noisy training procedure. They find that when the angle formed by the pre-trained model and the two models to be souped is larger, there is a greater performance boost from souping. A wider angle suggests that fine-tuned trajectories are more diverse, thus more variance is reduced by averaging. Understanding why such averaging remains in low-loss regions, how diversity among fine-tuned models contributes to robustness, and when souping is beneficial is therefore essential for studying adaptation more broadly.

1.1 RELATED WORK

Souping: Souping has been used in a variety of settings. Croce et al. (2023) soup models trained to be robust to different distribution shifts and Ramé et al. (2023) soup ingredients trained on different tasks. Both cases lead to better generalisation. Jang et al. (2025) further explore the optimal soup between just two fine-tuned models by considering the angle formed by the two training trajectories in a layer-wise fashion.

Stochastic Weight Averaging (SWA): Souping is related to the idea of SWA Izmailov et al. (2019). 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. **Exponential Moving Average (EMA)**: Another method for averaging weights over training is EMA Morales-Brotons et al.

(2024). Here, the weights at each training step are combined with the previous average using an exponential decay.

Stability Analysis: Souping success requires models to be 'compatible' in the sense that averaging their weights has low loss. Frankle et al. (2020) define *stability to SGD noise*, whereby at some point during training, models become robust to the noise from SGD in the sense that all possible minima obtained by training from that point onwards lie in the same low-loss basin.

1.2 Our Contributions

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

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.
Can we predict the effectiveness of souping using similarity measures? If two models are very similar, it may be more likely that they can soup effectively.
Is souping transitive — If model A soups with B, and B soups with C, will A soup with C?
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.

Further experiments investigating the effect of permuting models Ainsworth et al. (2023) prior to souping and how souping affects robustness can be found in Appendix A.1 and A9 respectively.

1.3 Soups and Soup Gain

For this work, we only soup pairs of models θ_A and θ_B using the simple arithmetic mean. That is, $\theta_{\text{soup}} = \frac{1}{2}\theta_A + \frac{1}{2}\theta_B$. We do not consider any other weighted average in order to save on computational cost. Work by Ainsworth et al. (2023), shows many of the loss barriers they find have the most extreme behaviour at the midpoint. Thus we assume that the midpoint serves as an accurate summary statistic of souping performance over all possible convex combinations.

We define the *soup gain* of a pair of models θ_A, θ_B as the decrease in loss over a test set obtained by souping the models. That is, soup gain $= \min\{L(\theta_A), L(\theta_B)\} - L(\theta_{\text{soup}})$ where $L(\theta)$ denotes the test loss obtained using model θ. Soup gain can also be computed in terms of accuracy rather than loss. We use the soup gain as the primary measure of the effectiveness of souping in our experiments. We prefer to compare to the minimum rather than the mean of the ingredients as the purpose of a soup should be to improve over its ingredients.

2 Experiments

2.1 Method

We train a baseline model for image classification on the CIFAR-100 dataset with ResNet-50 He et al. (2016) using a well-known set of hyper-parameters Dadalto (2023) with image reflection and random translation with padding for data augmentation. We hold out 5% of the $50,000$ training images as a validation set, as well as $10,000$ images for testing. The baseline model is saved every 10 epochs. From each save point, we branch off and train 4 new models with different optimizer settings. For details, see Appendix Tables 1 and 2. All models are each trained to convergence, with the best validation scored model weights saved for the experiments. This process is illustrated in Figure A2. A total of 4 variants and 26 branch points were trained, yielding 104 related models and $5,356$ binary souping combinations for analysis.

2.2 Shared Epochs and Soup Gain

We group the soups by the number of shared epochs between the two ingredients before they diverge into different training trajectories. For example, a pair of models branched form the baseline at epoch 50 and 100 respectively share 50 epochs. In Figure 1, we plot quantiles of soup gain as a function of shared epochs. We observe the distribution of soup gains shifts positively as the number of shared epochs increases. However, the soup gain is often large and negative until around epoch 150. After epoch 250, nearly all soups are approximately neutral.

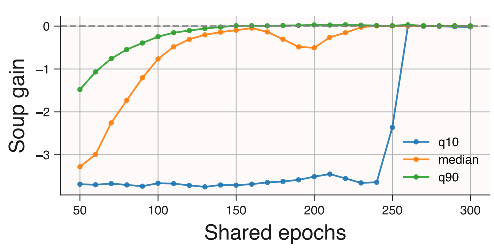

Figure 1: Quantiles of soup gain vs shared epochs.

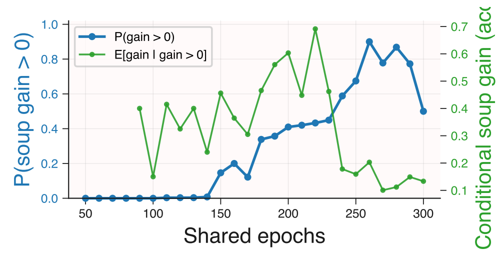

Figure 2: Probability of positive soup gain and conditional expected gain vs shared epochs.

To better illustrate how much souping helps, we plot the probability of positive soup gain and the average soup gain in accuracy for such positive soups in Figure 2.The probability of positive soup gain increases with shared epochs, reaching around 80% after 250 shared epochs. The conditional expected soup gain is noisy, at around 0.5% accuracy improvement when souping works. Towards the upper end of shared epochs, the gains decrease to around 0.2%. This suggests that while souping becomes more likely to work with increased shared training, the magnitude of gain decreases when the models are too similar.

2.3 Predicting Soupability with Similarity

Given two models, can we predict whether or not they will soup? To test this hypothesis, we compute a variety of similarity and distance metrics between pairs of models. We find that all metrics perform similarly. A plot for all metrics can be found in Figure A3. We arbitrarily choose to show the KL divergence between the outputs of the ingredients in Figure 3 as an example. There is a strong correlation between KL divergence and soup gain with a Spearman correlation between of -0.86. However, many of these soups have very poor performance. Figure 4 shows the soup gain against KL divergence for only those models with positive soup gain, with a moderate positive correlation with a Spearman correlation of 0.39. Models must be sufficiently similar in order to soup, but to be effective, the models must also be sufficiently different. Balancing these two effects is key to tasty soups.

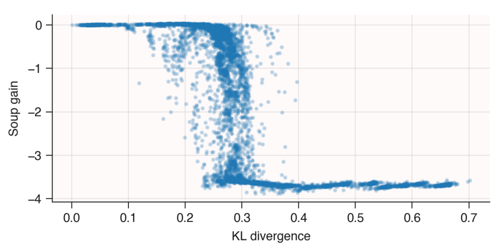

Figure 3: KL vs soup gain (Spearman -0.86).

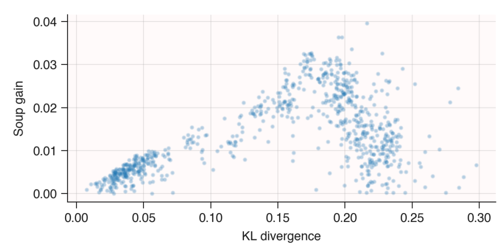

Figure 4: KL vs soup gain, positive soups only (Spearman 0.39).

2.4 Is Souping Transitive?

If model A soups successfully with models B and C, will B and C also soup successfully? If models soups together when they lie in the same low loss region, then transitivity should hold. To test this hypothesis, we consider all triplets of models (A, B, C). We plot the probability that B and C soup against the number of positive soups involving A in Figure 5. We observe that the probability that B and C soup is only high when A soups with both B and C. We also plot the soup gain between B and C against the minimum soup gain of (A, B) and (A, C) in Figure A5, noting a moderate positive Spearman correlation of 0.64.

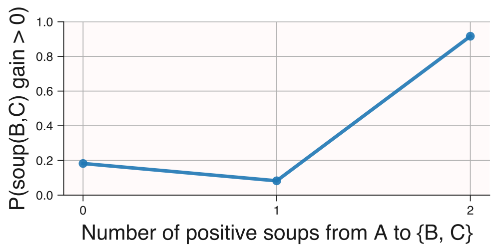

Figure 5: Probability of positive soup gain of B and C vs positive soups with A.

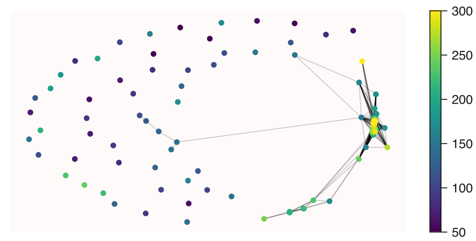

Figure 6: 2D embedding of 104 models using soup-gain distance; edges indicate positive soups.

Following transitivity, we investigate whether there are clusters of models that soup well together. We embed our 104 models into 2D using soup gain as a distance metric and plot them in Figure 6. The details of generating this plot can be found in Section A.4. We find that most successful soups lie in a single cluster of models all branched from later on in the training procedure. We conclude that souping is approximately transitive. There are many counter-examples, but this work supports the idea that, in general, soups lie in a single loss basin.

2.5 Souping for Robustness to Corruption

All experiments thus far have measured soup gain on a held-out test set. However, souping has also been used for robustness to distribution shift (Croce et al., 2023). To establish whether souping for in-distribution performance also increases out-of-distribution performance, we compute the soup gain on CIFAR-100C (Hendrycks & Dietterich, 2019) with severity level 3. The soup gains on test and corrupted data correlate very well, with a Spearman correlation of 0.99. A scatterplot can be found in Figure A6. The positive trend still holds when conditioning on only soups with positive soup gain on the test set, with a Pearson correlation of 0.61. A plot can be found in Figure A8. In-distribution performance improvement transfers to unseen target distributions.

We also plot the probability of positive soup gain on corrupted data as a function of shared epochs in Figure A7. 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.

3 Conclusion

In this work, we have conducted a series of experiments to further our understanding of souping. We find that ingredients must have sufficiently many shared epochs of training in order to be compatible but that too many shared epochs and the soup gain is minimal. We also find that various similarity measures between models correlate similarly with soup gain. Similar ingredients are less likely to collapse when souped, but very similar ingredients 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. 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 variance reduction and consequently the reduction in loss we expect from souping.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries, 2023. URL https://arxiv.org/abs/2209.04836.

Francesco Croce, Sylvestre-Alvise Rebuffi, Evan Shelhamer, and Sven Gowal. Seasoning model soups for robustness to adversarial and natural distribution shifts, 2023. URL https://arxiv.org/abs/2302.10164.

Eduardo Dadalto. Resnet-50 model trained on cifar-100. https://huggingface.co/edadaltocg/resnet50_cifar100, 2023. Hugging Face Model Repository.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis, 2020. URL https://arxiv.org/abs/1912.05671.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778, Las Vegas, NV, USA, June 2016. IEEE. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations, 2019. URL https://arxiv.org/abs/1903.12261.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2019. URL https://arxiv.org/abs/1803.05407.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models, 2025. URL https://arxiv.org/abs/2403.19522.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits, 2024. URL https://arxiv.org/abs/2411.18704.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization, 2023. URL https://arxiv.org/abs/2212.10445.

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. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 2022. URL https://arxiv.org/abs/2203.05482.

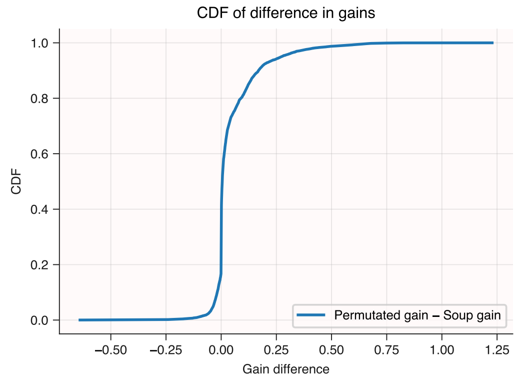

Figure A1: Cumulative distribution function (CDF) of the difference in soup gain before and after permutation alignment using `rebasin`. This ignores the 7% of soups with a loss higher than 5 after permutation as these make the plot difficult to interpret. The remaining mean and median difference is 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.

A Appendix

A.1 Permutation Alignment for Souping

Following the work from Ainsworth et al. (2023), 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/`

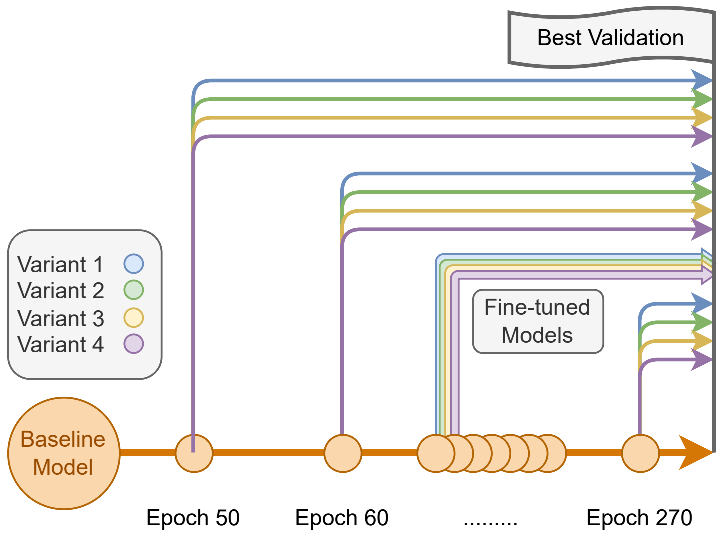

Figure A2: Branching of fine-tuned models from baseline checkpoints. A single baseline model is trained, with checkpoints saved every 10 epochs. From each checkpoint, 4 variants are trained with different optimizer hyper-parameter perturbations.

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

A.2 Training Details for CIFAR-100 with ResNet-50

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 2: Training hyperparameters for CIFAR-100 with ResNet-50.

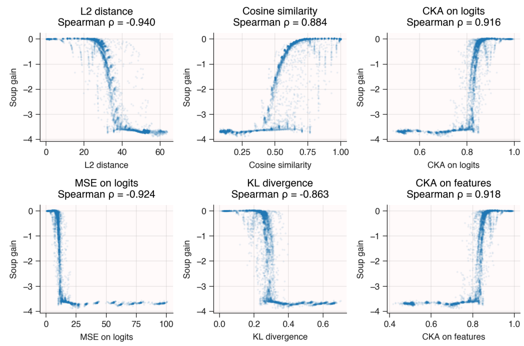

Figure A3: Soup gain vs various similarity and distance metrics between pairs of models. Each subplot shows the soup gain against one metric, with the Spearman correlation. All metrics perform similarly. We conclude that the more similar models are, the more likely souping is to not cause model collapse.

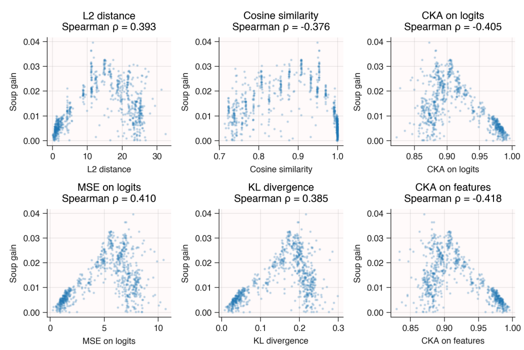

Figure A4: Soup gain vs various similarity and distance metrics between pairs of models, for the subset of soups with positive soup gain. Each subplot shows the soup gain against one metric, with the Spearman correlation. Each metric correlates similarly with soup gain. We see that when models are very similar, soup gain is small, while more dissimilar models can yield larger soup gains.

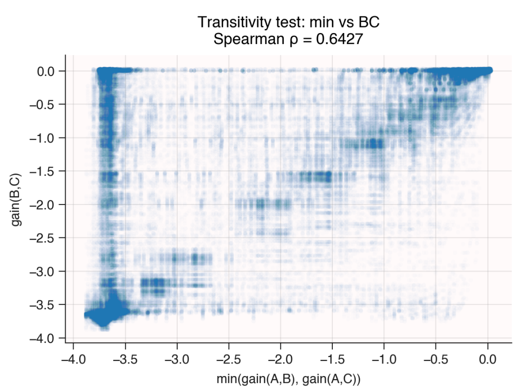

Figure A5: Scatterplot of the soup gain of models B and C against the minimum soup gain of models A with B and C. Each point represents a triplet of models (A, B, C). We observe a positive Spearman correlation of 0.64, suggesting that souping is fuzzily transitive. The correlation with the mean soup gain is lower, at 0.49.

Figure A6: 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.

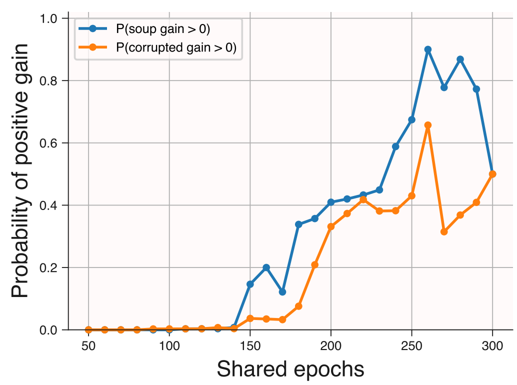

Figure A7: Probability of positive gain for soups as a function of the number of shared epochs. We see that the probability of positive gain increases with the number of shared epochs, for both clean and corrupted data. However,the corrupted data consistently has a slightly lower probability of positive gain. Thus, while souping also helps on corrupted data, it is slightly less effective than on clean data.

A.3 Further Details on Experiments

A.4 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 6.

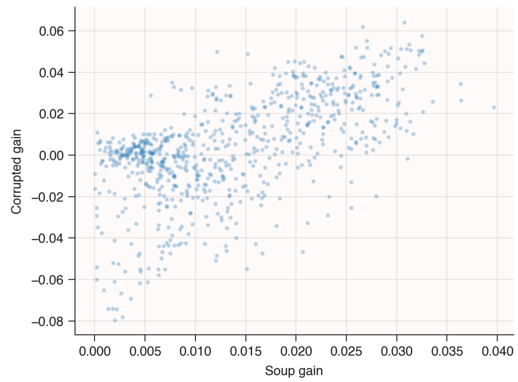

Figure A8: Plot of soup gain on test vs corrupted data for only models with positive soup gain on the test set. Spearman correlation of 0.61.

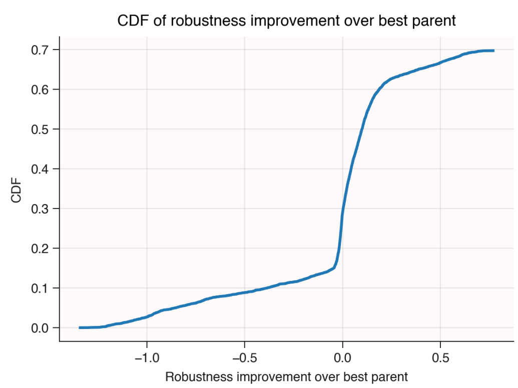

Figure A9: CDF of the difference in *robustness gap* before and after souping. The robustness gap is defined as the difference in loss between the test set and the corrupted set. The robustness gap before souping is taken as the minimum robustness gap of the parents. We see a fairly symmetric distribution. It has mean -0.04 and median 0.02. We therefore conclude that souping does not systematically improve the robustness gap.