Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time

Mitchell Wortsman[†] Gabriel Ilharco[†] Samir Yitzhak Gadre[◇] Rebecca Roelofs[‡]

Raphael Gontijo-Lopes[‡] Ari S. Morcos[○] Hongseok Namkoong[◇] Ali Farhadi[†]

Yair Carmon[*§] Simon Kornblith[*‡] Ludwig Schmidt[*†]

Abstract

The conventional recipe for maximizing model accuracy is to (1) train multiple models with various hyperparameters and (2) pick the individual model which performs best on a held-out validation set, discarding the remainder. In this paper, we revisit the second step of this procedure in the context of fine-tuning large pre-trained models, where fine-tuned models often appear to lie in a single low error basin. We show that averaging the weights of multiple models fine-tuned with different hyperparameter configurations often improves accuracy and robustness. Unlike a conventional ensemble, we may average many models without incurring any additional inference or memory costs—we call the results "model soups." When fine-tuning large pre-trained models such as CLIP, ALIGN, and a ViT-G pre-trained on JFT, our soup recipe provides significant improvements over the best model in a hyperparameter sweep on ImageNet. As a highlight, the resulting ViT-G model attains 90.94% top-1 accuracy on ImageNet, a new state of the art. Furthermore, we show that the model soup approach extends to multiple image classification and natural language processing tasks, improves out-of-distribution performance, and improves zero-shot performance on new downstream tasks. Finally, we analytically relate the performance similarity of weight-averaging and logit-ensembling to flatness of the loss and confidence of the predictions, and validate this relation empirically.

1 Introduction

In recent years, research has shown that models pre-trained on large and diverse datasets learn representations that transfer well to a variety of tasks. As a result, machine learning practitioners now commonly develop solutions for downstream tasks by fine-tuning large pre-trained models [38, 100, 51, 50]. Typically, the fine-tuning process involves two steps: (1) fine-tune models with a variety of hyperparameter configurations, and (2) select the model which achieves the highest accuracy on the held-out validation set. The remaining models are then discarded.

Selecting a single model and discarding the rest has several downsides. For one, the selected model may not achieve the best performance. In particular, ensembling outputs of many models can outperform the best single model, albeit at a high computational cost during inference. For another, fine-tuning a model on downstream tasks can sometimes reduce out-of-distribution performance [72, 2, 96, 69].

[*]Equal contribution, authors ordered alphabetically.
[†]University of Washington [◇]Columbia University [‡]Google Research, Brain Team [○]Meta AI Research [§]Tel Aviv University

Method	ImageNet acc. (top-1, %)	Distribution shifts
ViT-G [102]	90.45	–
CoAtNet-7 [22]	90.88	–
Our models/evaluations based on ViT-G:		
ViT-G (reevaluated)	90.47	82.06
Best model in hyperparam search	90.78	84.68
Greedy soup	**90.94**	**85.02**

Table 1: *Model soups* improve accuracy over the best individual model when fine-tuning a JFT-3B pre-trained ViT-G/14 model [102] on ImageNet. Instead of selecting the best model from a hyperparameter sweep during fine-tuning, *model soups* average the weights of multiple fine-tuned models. To evaluate model performance under distribution shift we consider average accuracy on ImageNet-V2 [75], ImageNet-R [42], ImageNet-Sketch [90], ObjectNet [6], and ImageNet-A [43]. Additional results and details are provided by Table 4 and Section 3.3.2.

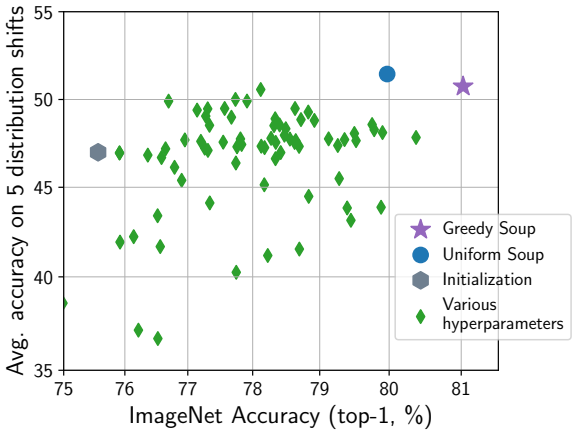

Figure 1: *Model soups* improve accuracy over the best individual model when performing a large, random hyperparameter search for fine-tuning a CLIP ViT-B/32 model on ImageNet. The *uniform soup* (blue circle) averages all fine-tuned models (green diamonds) in a random hyperparameter search over learning rate, weight-decay, iterations, data augmentation, mixup, and label smoothing. The *greedy soup* adds models sequentially to the model soup, keeping a model in the soup if accuracy on the held-out validation set does not decrease.

In this work, we propose a more accurate and robust alternative to the second step of the conventional recipe in the context of fine-tuning a large pre-trained model. Instead of selecting the individual fine-tuned model which achieves the highest accuracy on the held-out validation set, we average the weights of models fine-tuned independently, and refer to the result as a *model soup*. Given the results of the first step—a hyperparameter sweep over fine-tuned models—averaging several of these models to form a model soup requires no additional training and adds no cost at inference time.

Since neural networks are non-linear with potentially many solutions in different loss basins, it is perhaps surprising that averaging the weights of independently fine-tuned models achieves high performance. However, recent work [67] observes that fine-tuned models optimized independently from the same initialization lie in the same basin of the error landscape, inspiring our method. Weight averaging along a single training trajectory has previously been shown to improve the performance of models trained from random initialization [85, 46]. Our approach extends weight averaging to the context of fine-tuning, where we find that it also works across many independent runs.

We perform a comprehensive experimental study of fine-tuning to understand the behavior of model soups. For our main results we fine-tune CLIP [72] and ALIGN [47], which are pre-trained with a contrastive loss on image-text pairs, and a ViT-G model pre-trained on JFT [102]. Our results show that model soups often outperform the best individual model on both the in-distribution and natural distribution shift test sets (Table 1, Figure 1, Figure 4). A model soup composed of ViT-G models achieves 90.94% on ImageNet [23], surpassing the previous state of the art of 90.88% obtained by the CoAtNet model [22] while requiring 25% fewer FLOPs at inference time. In general, model soups can approach the performance of ensembling, with no additional computational cost or memory relative to a single model during inference. Beyond ImageNet and associated distribution shifts, our results show that model soups are applicable when fine-tuning on tasks from the WILDS [49] benchmark, and when fine-tuning transformer models [88, 24, 74] for text classification.

While the most straightforward approach to making a model soup is to average all the weights uniformly, we

Table 2: The primary methods contrasted in this work. Each θ_i is a model found through fine-tuning from a shared initialization. Cost refers to the memory and compute requirements during inference relative to a single model. All methods require the same training.

	Method	Cost
Best on val. set	$f(x, \arg\max_i \mathsf{ValAcc}(\theta_i))$	$\mathcal{O}(1)$
Ensemble	$\frac{1}{k}\sum_{i=1}^{k} f(x, \theta_i)$	$\mathcal{O}(k)$
Uniform soup	$f\left(x, \frac{1}{k}\sum_{i=1}^{k} \theta_i\right)$	$\mathcal{O}(1)$
Greedy soup	Recipe 1	$\mathcal{O}(1)$
Learned soup	Appendix B.1	$\mathcal{O}(1)$

find that *greedy soups*, where models are sequentially added to the soup if they improve accuracy on held-out data, outperforms uniform averaging. Greedy soups avoid adding in models which may lie in a different basin of the error landscape, which could happen if, for example, models are fine-tuned with high learning rates.

In addition to empirical observations, we analytically relate the similarity in loss between weight-averaging and logit-ensembling to the flatness of the loss (i.e., its second derivative on a line between models) and confidence of the predictions (expressed via the variance of a logits difference drawn from the weight-average softmax). We empirically validate our approximation on a subset of the models we train and show that it is strongly correlated with the true averaging vs. ensembling performance difference, particularly in the learning rate regimes where soups are effective and models achieve higher accuracy.

Paper outline. Our method of *model soups* is presented and evaluated in Sections 2 and 3, respectively. Next, Section 4 includes our analysis relating model soups and ensembles, Section 5 details the scope and limitations of the proposed method, and Section 6 contextualizes *model soups* by reviewing related work.

2 Method

This section highlights three recipes for model souping, the *uniform*, *greedy*, and *learned* soup, though the greedy soup is our central method. We summarize the methods described in this section in Table 2.

We consider a neural network $f(x, \theta)$ with input data x and parameters $\theta \in \mathbb{R}^d$. Fine-tuning is analogous to standard neural network training but includes an important distinction: the parameters are initialized to those found via pre-training.

Let $\theta = \mathsf{FineTune}(\theta_0, h)$ denote the parameters obtained by fine-tuning with pre-trained initialization θ_0 and hyperparameter configuration h. The hyperparameter configuration can include the choice of optimizer, data augmentation, training iterations, and a random seed which will determine data order.

For hyperparameter configurations $h_1, ..., h_k$ let $\theta_i = \mathsf{FineTune}(\theta_0, h_i)$. Conventionally, the parameters θ_j which attain the highest accuracy on a held out validation set are selected, and the remaining parameters are discarded. Instead, *model soups* $f(x, \theta_{\mathcal{S}})$ use an average of θ_i, i.e., $\theta_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} \theta_i$ where $\mathcal{S} \subseteq \{1, ..., k\}$. The *uniform soup* is constructed by averaging *all* fine-tuned models θ_i and so $\mathcal{S} = \{1, ..., n\}$.

There are settings in which a hyperparameter configuration can produce a model with low accuracy that results in a low accuracy uniform soup. This issue can be circumvented with a *greedy soup* (Recipe 1). The greedy soup is constructed by sequentially adding each model as a potential ingredient in the soup, and only keeping the model in the soup if performance on a held out validation set (disjoint from the training and test sets) improves. Before running this procedure we sort the models in decreasing order of validation set accuracy, and so the greedy soup can be no worse than the best individual model on the held-out validation set. We also explore a more advanced *learned soup* recipe that optimizes model interpolation weights by

Recipe 1 GreedySoup

Input: Potential soup ingredients $\{\theta_1, ..., \theta_k\}$ (optionally sorted in decreasing order of $\text{ValAcc}(\theta_i)$).
ingredients $\leftarrow \{\}$
for $i = 1$ **to** k **do**
  if $\text{ValAcc}(\text{average}(\text{ingredients} \cup \{\theta_i\})) \geq \text{ValAcc}(\text{average}(\text{ingredients}))$ **then**
    ingredients $\leftarrow$ ingredients $\cup \{\theta_i\}$
return ingredients

gradient-based minibatch optimization (see Appendix B.1 for details). This procedure requires simultaniously loading all models in memory which currently hinders its use with large networks.

3 Experiments

This section presents our key experimental findings. We begin with experimental setup (Section 3.1) and provide intuition for model soups by examining error landscape visualizations (Section 3.2). Next we present our main results (Section 3.3), using model soups as an alternative to selecting the best performing individual model. Finally, we explore model soups in the context of robust fine-tuning (Section 3.4), and examine model soups constructed by fine-tuning on different datasets (Section 3.5).

3.1 Experimental setup

Our experiments explore the application of model soups when fine-tuning various models. The primary models we fine-tune are the CLIP [72] and ALIGN [47] models pre-trained with contrastive supervision from image-text pairs, a ViT-G/14 model pre-trained on JFT-3B [102], and transformer models for text classification [24, 73]. Unless otherwise mentioned, experiments use the CLIP ViT-B/32 model. Fine-tuning is performed end-to-end (all parameters are modified) which often results in better accuracy than training only the final linear layer [51, 1, 17, 3].

We consider two different methods for initializing the final linear layer before fine-tuning. The first method initializes the model from a linear probe (LP), as described in Kumar et al. [54], and we refer to this method as LP initialization. The second method uses the zero-shot initialization, e.g., using the classifier produced by the text tower of CLIP or ALIGN as the initialization. Both methods for initializing the model produce similar trends when applicable, and unless otherwise stated we use the LP initialization.

For the ensemble baselines [26, 55] we ensemble the logits (unormalized outputs) of models as in Gontijo-Lopes et al. [39]. Fine-tuning uses a supervised cross-entropy loss and, unless otherwise mentioned, is conducted on ImageNet [23]. When fine-tuning on ImageNet we also evaluate on the five natural distribution shifts: ImageNetV2 [75], ImageNet-R [42], ImageNet-Sketch [90], ObjectNet [6], and ImageNet-A [43]. We often report results averaged over these five distribution shifts. Since the official ImageNet validation set is used as the test set, we use roughly 2% of the ImageNet training set as a held-out validation set for constructing greedy soups.

3.2 Intuition with error landscape visualizations

To provide intuition, we visualize a two dimensional slice of the training loss and test error landscape when fine-tuning CLIP on ImageNet. In these experiments, we use the zero-shot initialization $\theta_0 \in \mathbb{R}^d$ and fine-tune twice, independently, to produce solutions θ_1 and θ_2. The points θ_0, θ_1 and θ_2 define a plane in parameter space, and we evaluate the ImageNet train loss, ImageNet test error, and the test error on the five aforementioned distribution shifts on this plane. The results are illustrated in Figure 2 where the zero-shot initialization (θ_0) is shown as a star and a solution fine-tuned with learning rate $3 \cdot 10^{-5}$ (θ_1) is shown as a blue square. For θ_2 we either use the same learning rate as θ_1 (but vary the random seed) or learning rate $3 \cdot 10^{-6}$.

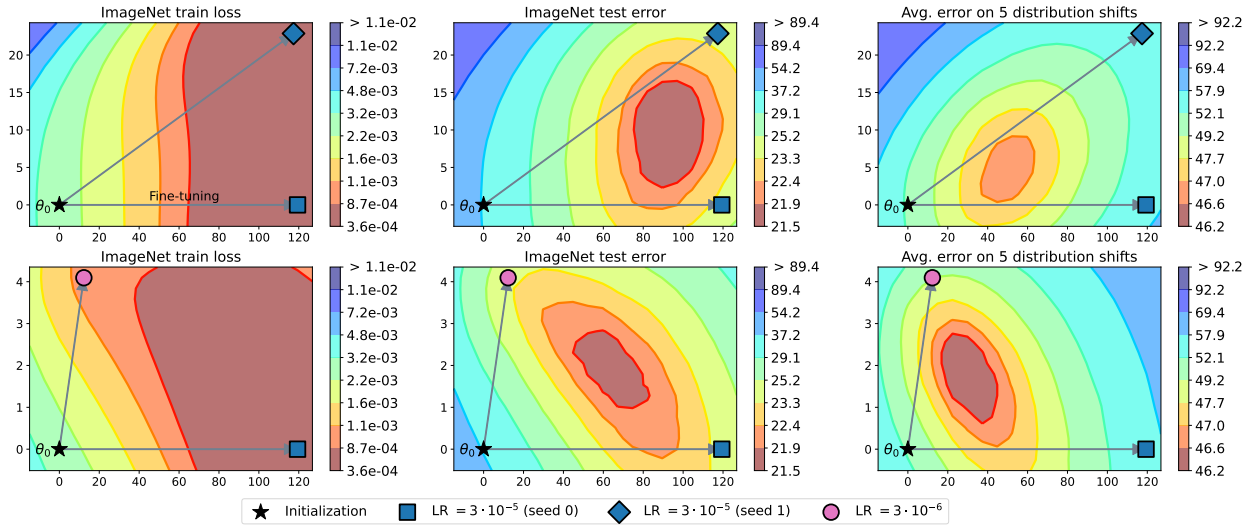

Figure 2: The solution with the highest accuracy is often not a fine-tuned model but rather lies between fine-tuned models. This figure shows loss and error on a two dimensional slice of the loss and error landscapes. We use the zero-shot initialization θ_0 and fine-tune twice (illustrated by the gray arrows), independently, to obtain solutions θ_1 and θ_2. As in Garipov et al. [36], we obtain an orthonormal basis u_1, u_2 for the plane spanned by these three models, and the x and y-axis show movement in parameter space in these directions, respectively.

For both the in-distribution and out-of-distribution test sets, the loss/error contours are basin-shaped, and none of the three points is optimal.

These results suggest that (1) interpolating the weights of two fine-tuned solutions can improve accuracy compared to individual models and (2) more uncorrelated solutions—models that form an angle[1] closer to 90 degrees—may lead to higher accuracy on the linear interpolation path.

To investigate the correlation between accuracy improvement and angle, we consider a series of models trained with different seeds, learning rates, and data augmentation. For each pair θ_1, θ_2, we compare the accuracy of their average with the average of their accuracies, $\mathsf{Acc}\left(\frac{1}{2}\theta_1 + \frac{1}{2}\theta_2\right) - \frac{1}{2}\left(\mathsf{Acc}\left(\theta_1\right) + \mathsf{Acc}\left(\theta_2\right)\right)$, which we refer to as the interpolation advantage. Figure 3 illustrates the results, in which we observe that the interpolation advantage is correlated with the angle ϕ and that varying the learning rate, seed, or data augmentation can produce solutions which are more orthogonal. Experimental details and discussion of high learning rates provided in Appendix B.2.

Finally, in Appendix C we ask the question: for a one dimensional grid of hyperparameters $\{h_a, ..., h_b\}$, how does averaging the models fine-tuned with hyperparameter configurations h_a and h_b corresponding to the endpoints compare with picking the best individual model fine-tuned with hyperparameter configuration $h \in \{h_a, ..., h_b\}$? The hyperparameters we vary are optimizer, augmentation, and learning rate. For the vast majority of grid searches, the average of the endpoints outperforms the best individual model in the grid.

3.3 Model soups

With the gains of averaging two fine-tuned models in mind, we turn our attention to averaging *many* models with different hyperparameters: this section presents our main results, which show that averaging fine-tuned models can be used as an alternative to the conventional procedure of selecting the single model which performs best on the held-out validation set. We explore CLIP [72] and ALIGN [47] fine-tuned on ImageNet [23] and WILDS [49] (Section 3.3.1), ViT-G pre-trained on JFT-3B [102] and fine-tuned on ImageNet

[1]In particular, the angle ϕ between $\theta_1 - \theta_0$ and $\theta_2 - \theta_0$, i.e., the angle between the arrows shown in Figure 2.

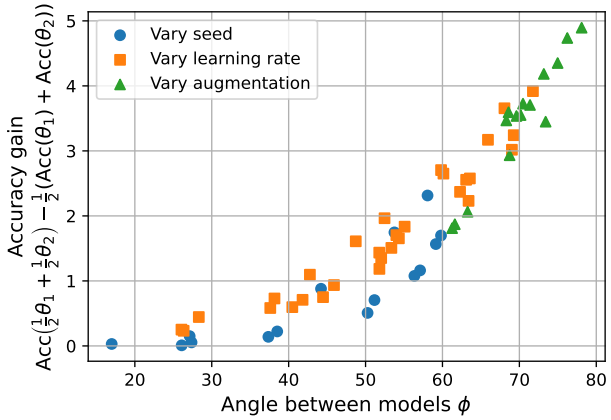

Figure 3: The advantage of averaging solutions (y-axis) is correlated with the angle ϕ between between solutions, while varying hyperparameter configurations between pairs enables a larger ϕ. Each point corresponds to a pair of models θ_1, θ_2 that are fine-tuned independently from a shared initialization θ_0 with different hyperparameter configurations. The angle ϕ between between solutions refers to the angle between $\theta_1 - \theta_0$ and $\theta_2 - \theta_0$ (i.e., the initialization is treated as the origin). Accuracy is averaged over ImageNet and the five distribution shifts described in Section 3.1.

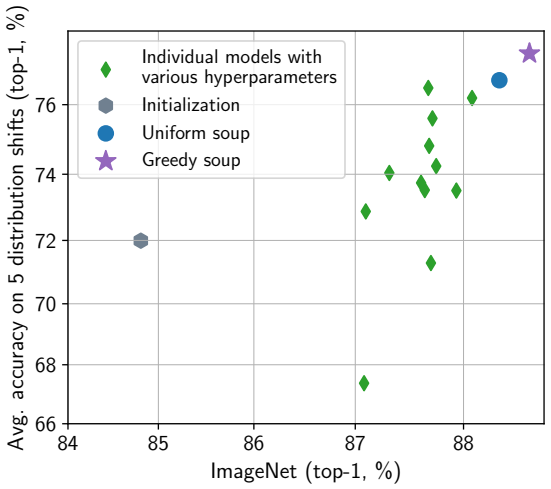

Figure 4: *Model soups* improve accuracy when fine-tuning ALIGN.

(Section 3.3.2), and transformer models fine-tuned on text classification tasks (Section 3.3.3). Appendix E additionally explores CLIP ViT-L fine-tuned on CIFAR-10 and ImageNet-22k-pretrained ViT-B/32 fine-tuned on ImageNet.

3.3.1 Fine-tuning CLIP and ALIGN

We begin our study of model soups by considering two-pretrained models, CLIP ViT-B/32 and ALIGN EfficientNet-L2, and performing a hyperparameter sweep for the fine-tuning each model on ImageNet. For CLIP we use a random hyperparameter search over learning rate, weight decay, training epochs, label smoothing, and data augmentation, obtaining 72 fine-tuned models (details in Appendix B.3.1). For ALIGN we use a grid search over learning rate, data augmentation, and mixup, obtaining 12 fine-tuned models (details in Appendix B.3.2). To form our greedy soups, we sort models in order of decreasing accuracy on the held-out validation set before applying Recipe 1. For both CLIP and ALIGN, the greedy soup selects 5 models. Figure 1 and 4 show the performance of the resulting models and their uniform and greedy soups for CLIP and ALIGN. The greedy soup improves over the best model in the hyperparameter sweep by 0.7 and 0.5 percentage points, respectively.

Furthermore, we show that the greedy soup requires less models to reach the same accuracy as selecting the best individual model on the held-out validation set. We consider an additional setting where we prepare a sequence of soups by sequentially adding CLIP models from the hyperparameter sweep in random order. Figure 5 shows the performance of the uniform and greedy soup, as well as the best single model so far and a logit ensemble, as a function of the number of models considered. For essentially any number of models, the greedy soup outperforms the best single model on both the ImageNet and out-of-distribution test sets; the greedy soup is better than the uniform soup on ImageNet and comparable to it out-of-distribution. The logit ensemble is better than the greedy soup on ImageNet, but worse out-of-distribution.

Table 3 lists the performance of the CLIP soups and baselines described above, as well as additional soup

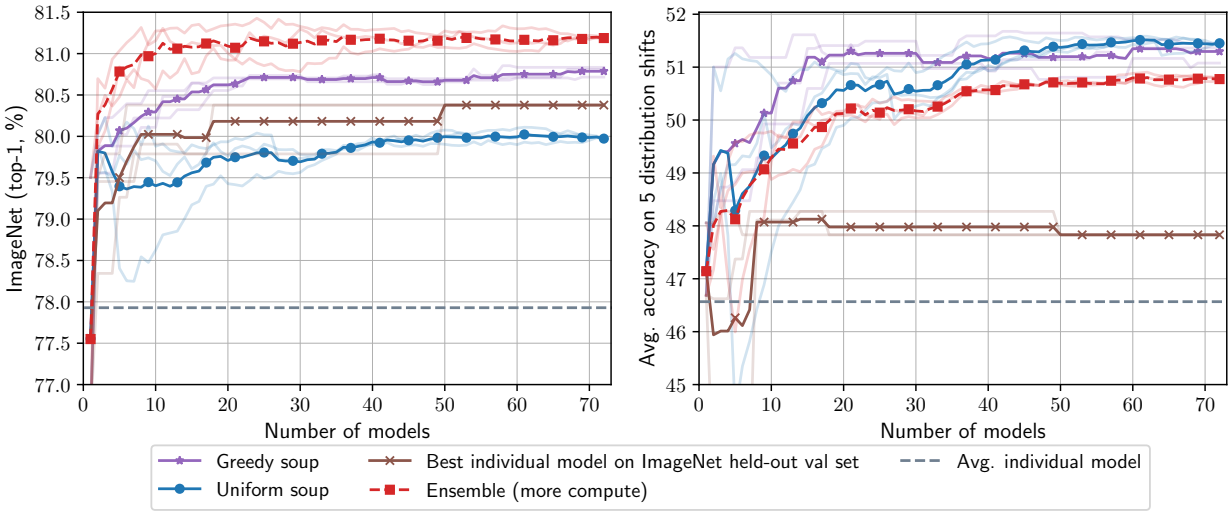

Figure 5: The greedy soup requires less models to reach the same accuracy as selecting the best individual model on the held-out validation set. On the x-axis we show the number of models considered in a random search over hyperparameters while the y-axis displays the accuracy of various methods for model selection which are summarized in Table 2. All methods require the same amount of training and compute cost during inference with the exception of the ensembles, which require a separate pass through each model. Results are for fine-tuning CLIP ViT-B/32, averaged over three random orders (shown with faded lines).

Table 3: Ablation on multiple methods from Table 2 and their variants when when fine-tuning CLIP ViT-B/32 with the random hyperparameter search described in Section 3.3.1. For "Greedy soup (random order)", we try three random orders report mean and standard deviation. The "Learned soup" and its variants are descried in Appendix B.1.

	ImageNet acc.	Distribution shifts acc.
Best individual model on ImageNet	80.38	47.83
Second best individual model on ImageNet	79.89	43.87
Uniform soup	79.97	51.45
Greedy soup (decreasing order of held-out val accuracy)	81.03	50.75
Greedy soup (random order)	80.79 (0.05)	51.30 (0.16)
Learned soup	80.89	51.07
Learned soup (by layer)	81.37	50.87
Ensemble	81.19	50.77
Greedy ensemble	81.90	49.44

variants described in Appendix B.1.

To further establish the generality of the model soup, we replicate the CLIP hyperparameter sweep experiment on two image classification tasks from WILDS [49], namely FMoW [18] and iWildCam [8]; Figure 6 shows results qualitatively similar to our ImageNet experiment, and Appendix B.3.1 describes experimental details.

We report several additional variants and baselines for the experiment described above. In Appendix D we present results for different hyperparameter sweeps and fine-tuning initializations, when fine-tuning CLIP on ImageNet. For instance, we try a *standard grid* search which is similar to the grid search described for ALIGN above, and an *extreme grid* search which includes solutions fine-tuned with extreme hyperparameters that result in badly performing models (details in Appendix B.3.1). Moreover, Appendix G compares model soups with additional baselines, including distillation from an ensemble as in [44], models which apply weight-averaging along their trajectory, and sharpness aware minimization [31].

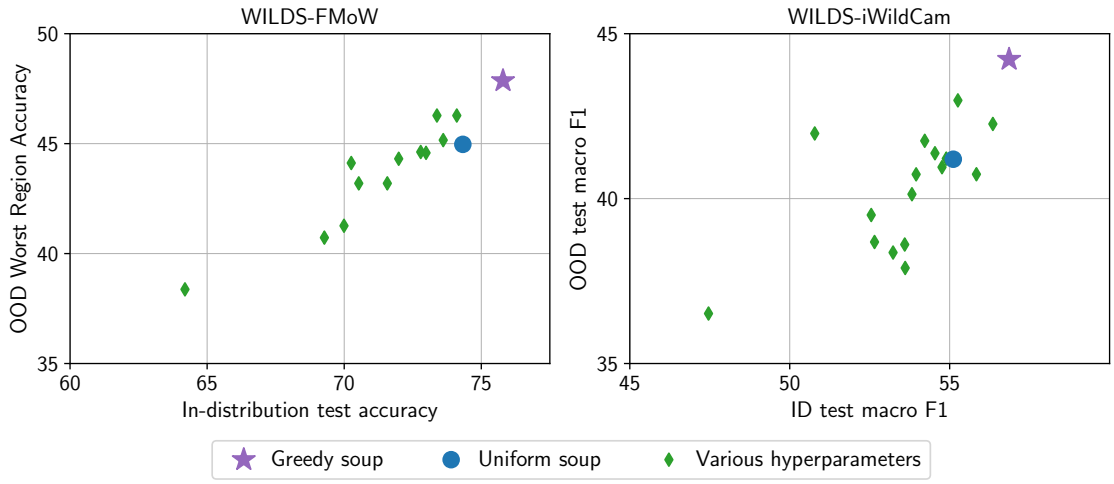

Figure 6: Model soups improve accuracy when fine-tuning on the diverse classification tasks WILDS-FMoW [49, 18] and WILDS-iWildCam [49, 8]. Results are shown for the CLIP ViT-L/14 model and a random hyperparameter search over learning rate, weight-decay, iterations, data augmentation, mixup, and label smoothing.

We highlight a few interesting takeaways from these experiments: (1) The greedy soup outperforms the best individual model—with no extra training and no extra compute during inference, we were able to produce a better model. (2) While the uniform soup can outperform the best individual model, we only observe this when all individual models achieve high accuracy (e.g., when fine-tuning ALIGN in Figure 1); unlike the examples in Figure 2, there can be an error barrier between fine-tuned models. We mainly observe this when fine-tuning with high learning rates (this is illustrated in Appendix B.2, Figure B.1). However, these high learning rate models also have a lower accuracy, and are therefore excluded by the greedy soup.

3.3.2 Fine-tuning a ViT-G model pre-trained on JFT-3B

To test whether the gains obtained by model soups are additive with other techniques used to obtain state-of-the-art models, we applied our greedy soup technique to 58 ViT-G/14 models fine-tuned on ImageNet. We vary the learning rate, decay schedule, loss function, and minimum crop size in the data augmentation, and optionally apply RandAugment [20], mixup [103], or CutMix [101]. We also train four models with sharpness-aware minimization (SAM) [31]. For further details of our hyperparameter sweep, see Appendix B.3.3. For each model training run, we save exponential moving averages (EMA) of the weights [85] computed with decay factors of 0.999 (low EMA) and 0.9999999 (high EMA). Whereas high EMA generally provides the best accuracy over the course of an individual training run, both greedy soup and greedy ensembling obtain higher validation accuracy when applied to parameters with low EMA. We report the highest single model accuracy numbers obtained with either EMA decay value, but perform greedy soup and ensembling with models trained with EMA decay of 0.999. For each combination of training run and EMA decay rate, we evaluate accuracy on our held out validation set every 1000 steps. We use these accuracy values to pick the best checkpoint for ensembling, souping, and subsequent evaluation.

In Table 4, we report results on the ImageNet validation set and the five distribution shift datasets studied above as well as two relabeled ImageNet validation sets, ReaL [10] and multilabel [79]. Our greedy soup procedure selects 14 of the 58 models fine-tuned as part of our hyperparameter sweep, and this soup performs statistically significantly better than the best individually fine-tuned model selected based on our held out validation set on all datasets except for ObjectNet. Even when we give an unfair advantage to individually fine-tuned models by selecting them based on their performance on each test set (denoted "oracle" in Table 4), the greedy soup, which was selected using only in-distribution data, remains superior on most datasets. Only

on ReaL and ObjectNet does there exist an individual model that performs statistically significantly better than the soup, and the best model differs between those two datasets. Greedy ensembling performs similarly to the greedy soup in terms of ImageNet top-1 and multilabel accuracy, and slightly better on ReaL, but significantly worse on all distribution shift datasets except for ImageNet-V2. Thus, greedy soup can provide additional gains on top of standard hyperparameter tuning even in the extremely high accuracy regime.

Table 4: Greedy soup improves over the best individual models obtained in a hyperparameter sweep for ViT-G/14 pre-trained on JFT-3B and fine-tuned on ImageNet, both in- and out-of-distribution. Accuracy numbers not significantly different from the best are bold-faced. Statistical comparisons are performed using an exact McNemar test or permutation test at $\alpha = 0.05$. Avg shift accuracy of the best model on each test set is the best average accuracy of any individual model.

	ImageNet			Distribution shifts					
Method	Top-1	ReaL	Multilabel	IN-V2	IN-R	IN-Sketch	ObjectNet	IN-A	Avg shifts
ViT/G-14 [102]	90.45	90.81	–	83.33	–	–	70.53	–	–
CoAtNet-7 [22]	90.88	–	–	–	–	–	–	–	–
Our models/evaluations based on ViT-G/14:									
ViT/G-14 [102] (reevaluated)	90.47	90.86	96.89	83.39	94.38	72.37	71.16	89.00	82.06
Best model on held out val set	90.72	91.04	96.94	83.76	95.04	73.16	78.20	91.75	84.38
Best model on each test set (oracle)	90.78	**91.78**	**97.29**	**84.31**	95.04	73.73	**79.03**	92.16	84.68
Greedy ensemble	**90.93**	91.29	**97.23**	**84.14**	94.85	73.07	77.87	91.69	84.33
Greedy soup	**90.94**	91.20	**97.17**	**84.22**	**95.46**	**74.23**	78.52	**92.67**	**85.02**

3.3.3 Fine-tuning on text classification tasks

To test whether the gains obtained by model soups extend to domains beyond image classification, we conduct preliminary experiments with natural language processing (NLP). While more investigation is warranted to establish the applicability of model soups for NLP, we believe our experiments are a promising initial step. In particular, we fine-tune BERT [25] and T5 [74] models on four text classification tasks from the GLUE benchmark [89]: MRPC [28], RTE [21, 5, 37, 9], CoLA [91] and SST-2 [84], as in [27]. We use the standard metric for each dataset: average of accuracy and F_1 score for MRPC, accuracy for RTE, Matthews correlation for CoLA [63] and accuracy for SST-2. More details are provided in Appendix H.

We fine-tune 32 models for each dataset with a random hyper-parameter search over learning rate, batch size, number of epochs and random seed. Table 5 reports the corresponding metric on the validation set for BERT-base uncased [24] and T5-base [74]. Additional experimental details and results for more models are provided in Appendix I. While the improvements are not as pronounced as in image classification, the greedy soup can improve performance over the best individual model in many cases.

3.4 Robust fine-tuning

Wortsman et al. [96] introduce WiSE-FT, a method for improving the robustness of a model θ_1 which is fine-tuned from initialization θ_0 by linearly interpolating θ_1 and θ_0. An intriguing observation was that,

Table 5: Performance of model soups on four text classification datasets from the GLUE benchmark [89].

Model	Method	MRPC	RTE	CoLA	SST-2
BERT [25]	Best individual model	88.3	61.0	59.1	92.5
	Greedy soup	88.3 (+0.0)	61.7 (+0.7)	59.1 (+0.0)	93.0 (+0.5)
T5 [74]	Best individual model	91.8	78.3	58.8	94.6
	Greedy soup	92.4 (+0.6)	79.1 (+0.8)	60.2 (+0.4)	94.7 (+0.1)

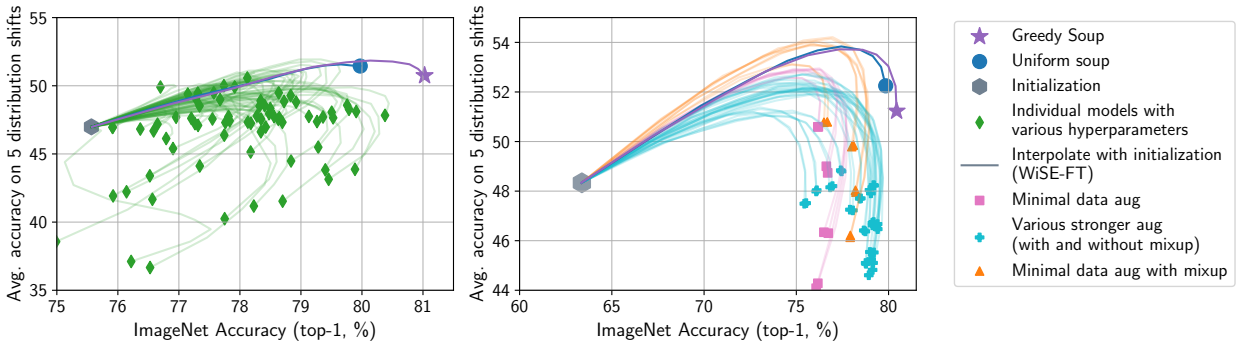

Figure 7: Model soups compared to baselines for robust fine-tuning. WiSE-FT [96] improves the robustness of model θ_1 fine-tuned from initialization θ_0 by interpolating between θ_1 and θ_0. Above we display the accuracy of models along these interpolation curves both for regular fine-tuned models and model soups (**left:** random hyperparameter search using the LP initialization, **right:** grid search using the zero-shot initialization). The model soups lie beyond the WiSE-FT curves generated by any individual model, and accuracy can be improved on the distribution shifts by applying WiSE-FT to the model soups.

once the data augmentation is fixed, interpolating between θ_1 and θ_0 often traces a similar curve regardless of hyperparameters.[2] In other words, a reasonable hypothesis was that this curve is Pareto optimal—no hyperparameter configuration would surpass it. In Figure 7, we trace the curves when interpolating between θ_1 and θ_0 for a random hyperparameter search (left) and the standard grid search described in Appendix B.3.1 (right) when fine-tuning CLIP ViT-B/32. We find that the uniform soup and greedy soup lie beyond these interpolation curves. Moreover, we find interpolating between these soups and the initialization also provides additional accuracy improvements on the distribution shifts.

3.5 Cross-dataset soups

So far, our experiments have studied soups of models fine-tuned on the same dataset with different hyperparameters. In this section, we prepare soups containing models fine-tuned on different datasets. We evaluate the resulting soups on a held-out dataset, from which no labeled training data is used (i.e., zero-shot evaluation).

Concretely, we consider soups based on the CLIP zero-shot initialization along with six models fine-tuned independently on CIFAR-10 [53], Describable Textures [19], Food-101 [13], SUN397 [97], Stanford Cars [52] and ImageNet [23]. We evaluate on CIFAR-100 [53], which does not share classes with CIFAR-10. Since each task has a different set of classes, the last layers cannot be part of the soup. Hence, during fine-tuning, we freeze the linear head produced by CLIP's text tower so that task-specific learning is captured only in the backbone weights. At test time, we use the "backbone soup" with a zero-shot head constructed from CLIP's text tower and the CIFAR-100 class names with the prompt-ensembling used for ImageNet by Radford et al. [72]. Figure 8 (left) shows that a model soup containing models trained on each of these datasets and the zero-shot model improves zero-shot performance on CIFAR-100 by 6.4 percentage points over the CLIP baseline. Moreover, Figure 8 (right) shows that the choice of which fine-tuned models to include can have a substantial impact on the accuracy of the resulting soup. See Appendix B.4 for additional details.

4 Analytically comparing soups to ensembles

The goal of this section is to obtain complementary analytical insight into the effectiveness of model soups. For simplicity, we consider a soup consisting of only two models with parameters θ_0 and θ_1. For weighting parameter $\alpha \in [0, 1]$ we let $\theta_\alpha = (1-\alpha)\theta_0 + \alpha\theta_1$ denote the weight-averaged soup. We would like to understand

[2] This is visible in Figure 7 (right) where different data augmentations are shown with different colors. On the other hand, in Figure 7 (left) there are many different methods of data augmentation as we conduct a random hyperparameter search.

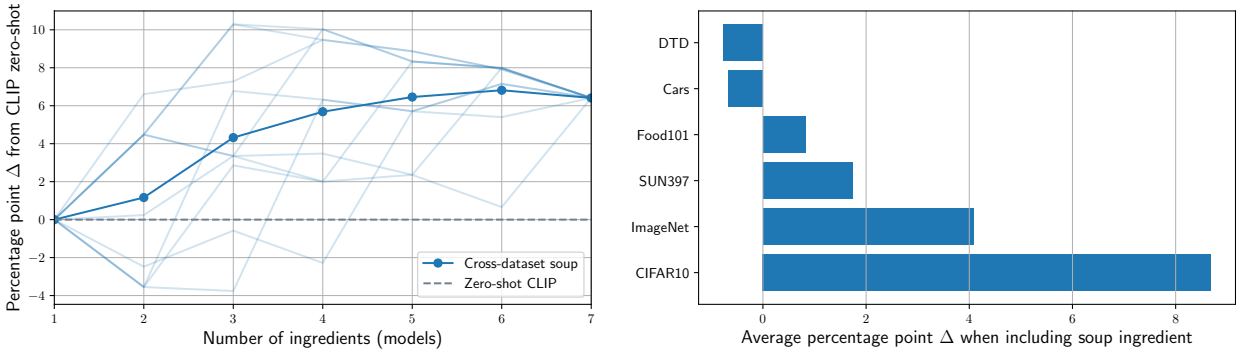

Figure 8: Model soups can improve zero-shot performance on new downstream tasks. **(left)** Starting with zero-shot CLIP we create a soup by adding models fine-tuned on ImageNet, CIFAR-10, Food101, SUN397, DTD, and Cars, and evaluate on CIFAR-100. Different orders for adding models are shown with faded lines. **(right)** The average change in CIFAR-100 accuracy when a model fine-tuned on the dataset listed in the y-axis is added to the model soup.

when would the soup error, $\text{err}_\alpha := \mathbb{E}_{x,y} 1\{\arg\max_i f_i(x;\theta_\alpha) \neq y\}$, be lower that the best of both endpoints, $\min\{\text{err}_0, \text{err}_1\}$.

Note that just convexity of err_α in α does not by itself imply superiority of the soup to both endpoints, as the minimum of err_α over α may be obtained at the endpoints even when err_α is convex. To get further leverage on the problem, we compare the soup to the *logit-level ensemble* $f^{\text{ens}}_\alpha(x) = (1-\alpha)f(x;\theta_0) + \alpha f(x;\theta_1)$. The rich literature on ensembles (see Sec. 6) tells us that the expected error of the ensemble, $\text{err}^{\text{ens}}_\alpha$, is often strictly below $\min\{\text{err}_0, \text{err}_1\}$ for neural networks. Therefore, whenever $\text{err}_\alpha \approx \text{err}^{\text{ens}}_\alpha$ we expect the soup to outperform both endpoint models.

To analytically compare the soup and the ensemble, we replace the 0-1 loss with a differentiable surrogate. Specifically, we the consider cross-entropy loss $\ell(f, y) = \log\left(\sum_{y'} e^{f_{y'} - f_y}\right)$. We let $\mathcal{L}^{\text{soup}}_\alpha = \mathbb{E}_{x,y}\ell(\beta f(x;\theta_\alpha), y)$ denote the β-calibrated expected loss of the soup, and similarly define $\mathcal{L}^{\text{ens}}_\alpha = \mathbb{E}_{x,y}\ell(\beta f^{\text{ens}}_\alpha(x), y)$ for the ensemble. We derive the following approximation for the loss difference:

$$\mathcal{L}^{\text{soup}}_\alpha - \mathcal{L}^{\text{ens}}_\alpha \approx \frac{\alpha(1-\alpha)}{2}\left(- \frac{\mathrm{d}^2}{\mathrm{d}\alpha^2}\mathcal{L}^{\text{soup}}_\alpha + \beta^2 \mathbb{E}_x \text{Var}_{Y\sim p_{\text{sftmx}}(\beta f(x;\theta_\alpha))}\left[\Delta f_Y(x)\right]\right), \tag{1}$$

where $[p_{\text{sftmx}}(f)]_i = e^{f_i}/\sum_j e^{f_j}$ is the standard "softmax" distribution and $\Delta f(x) = f(x;\theta_1) - f(x;\theta_0)$ is the difference between the endpoint logits. We obtain our approximation in the regime where the logits are not too far from linear; see Appendix F.3 for a detailed derivation.

The first term in approximation (1) is negatively proportional to the second derivative of the loss along the trajectory: when the approximation holds, convexity of the loss indeed favors the soup. However, the second term in the approximation does not follow from the "convex basin" intuition. This term always favors the ensemble, but is small in one of two cases: (a) the somewhat trivial case when the endpoint models are similar (so that Δf is small) and (b) when the soup produces confident predictions, implying that $p_{\text{sftmx}}(\beta f(x;\theta_\alpha))$ is close to a point mass and consequently the variance term is small.

To test our approximation, we evaluate it over of set of fine-tuned models with different learning rates, augmentation strategies, random seeds and α values. We set β to calibrate the soup model, and find that it improves the ability of our approximation to predict the soup/ensemble error difference; see Appendix F.4 for detailed description of our setup.

Figure 9 summarizes the results of our empirical evaluations. When excluding the high learning rate of 10^{-4}

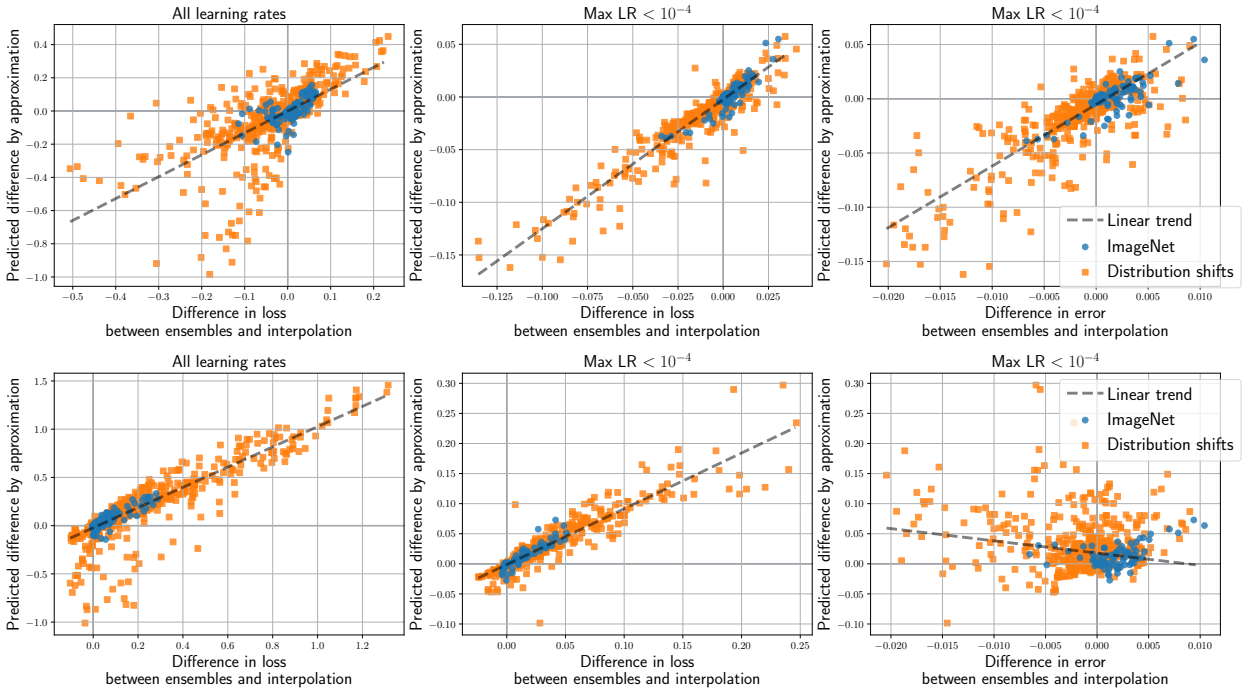

Figure 9: Validation of the analytical approximation (1) for the performance difference of a 2-model soup and ensemble. Each marker on the scatter plots represent a different choice of endpoint models (θ_0, θ_1) and interpolation weight α. In every scatter plot, the vertical axis shows the true performance difference between the soup and ensemble (in loss for the **left** and **center** panes, and error for the **right** pane), where a positive value indicates the ensemble is better. The horizontal axis shows our approximation for the loss difference. The **top** row shows results with inverse temperature β chosen to calibrate the soup, and the **bottom** row shows results for β fixed to 1.

(center and right panels),[3] we see that the approximation is strongly correlated with both the true difference in loss as well as the difference in error, and the approximation and true loss difference generally agree in sign. Additional details are provided in Appendix F.

5 Scope and limitations

While this work has so far demonstrated that averaging many fine-tuned models is a useful technique for improving accuracy, this section explores two limitations of the approach. The first is the applicability of model soups, and the second is the failure of model soups to substantially improve calibration.

Applicability. So far our experiments have mainly explored models pre-trained on large, heterogeneous datasets. In Appendix E we also explore model soups for an ImageNet-22k pre-trained model. While the greedy soup still provides improvements on ImageNet, these improvements are less substantial compared to those observed when fine-tuning CLIP and ALIGN.

Calibration. While ensembles improve model calibration [40, 76], model soups do not have the same effect. As hyperparameters can also have an effect on calibration, we consider the ensemble and soup of 20 models which are identical other than random seed. Results are illustrated in Figure A.1 using the calibration metrics of Roelofs et al. [76].

[3] Fine-tuned models with learning rate 10^{-4} are far in weight space from the initial model and are often rejected when forming greedy soups. Therefore, we do not expect our approximation to be tight for these learning rates.

6 Related work

Averaging model weights. Averaging the weights of models is a popular approach in convex optimization and deep learning. Most applications study models along the same optimization trajectory, e.g. [77, 70, 85, 46, 104]. By contrast, Frankle et al. [32], Neyshabur et al. [67] and Matena and Raffel [62] weight-average models which share an initialization but are optimized independently. Frankle et al. [32] find that, when training a pair of models from scratch with the same hyperparameter configuration but different data order, interpolating weights achieves no better than random accuracy. However, if the two models share a portion of their optimization trajectory, accuracy does not drop when they are averaged. Analogously, Neyshabur et al. [67] demonstrate that when two models are fine-tuned with the same pre-trained initialization, the interpolated model attains at least the accuracy of the endpoints. Unlike Frankle et al. [32] and Neyshabur et al. [67] we consider averaging many models with varied hyperparameter configurations.

Matena and Raffel [62] merge models with the same pre-trained initialization that are fine-tuned on different text classification tasks. They also propose Fisher information as an alternative technique for model merging. Unlike experiments in our Section 3.5, Matena and Raffel [62] use data from the target distribution for fine-tuning. Moreover, Wortsman et al. [96] average zero-shot and fine-tuned models, finding improvements in- and out-of-distribution. In contrast to Wortsman et al. [96], we average models across many independent runs which provides more substantial improvements in-distribution.

Stochastic Weight Averaging (SWA) [46], which averages weights along a single optimization trajectory, is also motivated by the relation between ensembling model outputs and averaging model weights. In contrast, the averaging we propose is across independent runs. Moreover, while their analysis relates the averaged network outputs (i.e., the logit ensemble) to the output of the a network with the averaged weights, our analysis (Section 4) goes a step further and relates the classification losses associated with these two vectors.

Pre-training and fine-tuning. In computer vision and natural language processing, the best performing models are often pre-trained on a large dataset before being fine-tuned on data from the target task [29, 100, 80, 38, 61, 51, 99, 50, 12]. This paradigm is also referred to as transfer learning. Recently, image-text pre-training has become increasingly popular in computer vision as a pre-training task [72, 47, 65, 69]. Recent work has explored alternative strategies for adapting these models to specific target tasks [106, 35, 105], for instance via a lightweight residual feature adapter. In contrast, our work explores standard end-to-end fine-tuned models. Other work has attempted to improve transfer learning by regularizing models toward their initialization [98], choosing layers to tune on a per-example basis [41], reinitializing layers over the course of training [57], or using multiple pretrained models with data-dependent gating [82].

Ensembles. Combining the outputs of many models is a foundational technique for improving the accuracy and robustness of machine learning models [26, 7, 14, 34, 55, 33]. Ovadia et al. [68] show that ensembles exhibit high accuracy under distribution shift. Mustafa et al. [66] propose a method for identifying subsets of pre-trained models for fine-tuning and later ensembling them, finding strong in-distribution accuracy and robustness to distribution shift. Gontijo-Lopes et al. [39] conduct a large-scale study of ensembles, finding that higher divergence in training methodology leads to uncorrelated errors and better ensemble accuracy. Finally, previous work has explored building ensembles of models produced by hyperparameter searches [83, 64, 78], including greedy selection strategies [15, 16, 56, 93]. Importantly, ensembles require a separate inference pass through each model, which increases computational costs. When the number of models is large, this can be prohibitively expensive. Unlike ensembles, model soups require no extra compute at inference time.

7 Conclusion

Our results challenge the conventional procedure of selecting the best model on the held-out validation set when fine-tuning. With no extra compute during inference, we are often able to produce a better model by averaging the weights of multiple fine-tuned solutions.

Acknowledgements

We thank Ting Chen, Jesse Dodge, Ben Eysenbach, David Fleet, Pieter-Jan Kindermans, Mohammad Norouzi, Sarah Pratt and Vivek Ramanujan for helpful discussions and draft feedback, Lucas Beyer and Xiaohua Zhai for assistance with ViT-G/14 fine-tuning, and Hyak at UW for computing support.

YC was supported in part by the Israeli Science Foundation (ISF) grant no. 2486/21, the Len Blavatnik and the Blavatnik Family foundation, and The Yandex Initiative for Machine Learning.

References

[1] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European conference on computer vision*, pages 329–344. Springer, 2014.

[2] Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The evolution of out-of-distribution robustness throughout fine-tuning, 2021. https://arxiv.org/abs/2106.15831.

[3] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 36–45, 2015.

[4] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018.

[5] Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proc. of the II PASCAL challenge*, 2006.

[6] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/97af07a14cacba681feacf3012730892-Paper.pdf.

[7] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 1999. https://link.springer.com/article/10.1023/A:1007515423169.

[8] Sara Beery, Arushi Agarwal, Elijah Cole, and Vighnesh Birodkar. The iwildcam 2021 competition dataset. In *Conference on Computer Vision and Pattern Recognition (CVPR) FGVC8 Workshop*, 2021. https://arxiv.org/abs/2105.03494.

[9] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[10] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[11] Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent, 2021. URL https://arxiv.org/abs/2106.05237.

[12] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models, 2021. https://arxiv.org/abs/2108.07258.

[13] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, 2014. https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/.

[14] Leo Breiman. Bagging predictors. *Machine learning*, 1996. https://link.springer.com/article/10.1007/BF00058655.

[15] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18, 2004.

[16] Rich Caruana, Art Munson, and Alexandru Niculescu-Mizil. Getting the most out of ensemble selection. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 828–833. IEEE, 2006.

[17] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[18] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. https://arxiv.org/abs/1711.07846.

[19] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. https://arxiv.org/abs/1311.3618.

[20] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. https://arxiv.org/abs/1909.13719.

[21] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005.

[22] Zihang Dai, Hanxiao Liu, Quoc Le, and Mingxing Tan. CoAtNet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009. https://ieeexplore.ieee.org/document/5206848.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. URL https://aclanthology.org/N19-1423.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[26] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 2000. https://link.springer.com/chapter/10.1007/3-540-45014-9_1.

[27] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[28] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*, 2005.

[29] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR, 2014.

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. https://arxiv.org/abs/2010.11929.

[31] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. https://openreview.net/forum?id=6Tm1mposlrM.

[32] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020. https://arxiv.org/abs/1912.05671.

[33] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997. https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[34] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*. Springer series in statistics New York, 2001.

[35] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021.

[36] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. https://arxiv.org/abs/1802.10026.

[37] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[39] Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods, 2021. https://arxiv.org/abs/2007.01434.

[40] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017. https://arxiv.org/abs/1706.04599.

[41] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4805–4814, 2019.

[42] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *International Conference on Computer Vision (ICCV)*, 2021. https://arxiv.org/abs/2006.16241.

[43] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. https://arxiv.org/abs/1907.07174.

[44] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Dark knowledge, 2014. https://www.ttic.edu/dl/dark14.pdf.

[45] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS) Deep Learning Workshop*, 2015. https://arxiv.org/abs/1503.02531.

[46] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. https://arxiv.org/abs/1803.05407.

[47] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2102.05918.

[48] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014. https://arxiv.org/abs/1412.6980.

[49] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2012.07421.

[50] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European Conference on Computer Vision (ECCV)*, 2020. https://arxiv.org/abs/1912.11370.

[51] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. https://arxiv.org/abs/1805.08974.

[52] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision (ICCV) Workshops*, 2013. https://ieeexplore.ieee.org/document/6755945.

[53] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[54] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=UYneFzXSJWh.

[55] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. https://arxiv.org/abs/1612.01474.

[56] Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian hyperparameter optimization for ensemble learning. *arXiv preprint arXiv:1605.06394*, 2016.

[57] Xingjian Li, Haoyi Xiong, Haozhe An, Cheng-Zhong Xu, and Dejing Dou. Rifle: Backpropagation in depth for deep transfer learning through re-initializing the fully-connected layer. In *International Conference on Machine Learning*, pages 6010–6019. PMLR, 2020.

[58] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2016. https://arxiv.org/abs/1608.03983.

[59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. https://openreview.net/forum?id=Bkg6RiCqY7.

[60] Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

[61] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision (ECCV)*, 2018. https://arxiv.org/abs/1805.00932.

[62] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging, 2021. https://arxiv.org/abs/2111.09832.

[63] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 1975.

[64] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In *Workshop on Automatic Machine Learning*, pages 58–65. PMLR, 2016.

[65] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training. *arXiv preprint arXiv:2112.12750*, 2021.

[66] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Deep ensembles for low-data transfer learning, 2020. https://arxiv.org/abs/2010.06866.

[67] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. https://arxiv.org/abs/2008.11687.

[68] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1906.02530.

[69] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong, Mingxing Tan, and Quoc V. Le. Combined scaling for zero-shot transfer learning, 2021. https://arxiv.org/abs/2111.10050.

[70] Boris Teodorovich Polyak. New method of stochastic approximation type. *Automation and remote control*, 1990.

[71] Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025.

[72] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. https://arxiv.org/abs/2103.00020.

[73] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020. http://jmlr.org/papers/v21/20-074.html.

[74] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[75] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning (ICML)*, 2019. https://arxiv.org/abs/1902.10811.

[76] Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C Mozer. Mitigating bias in calibration error estimation, 2020. https://arxiv.org/abs/2012.08668.

[77] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process, 1988. https://ecommons.cornell.edu/bitstream/handle/1813/8664/TR000781.pdf.

[78] Tonmoy Saikia, Thomas Brox, and Cordelia Schmid. Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*, 2020.

[79] Vaishaal Shankar, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. Evaluating machine accuracy on imagenet. In *International Conference on Machine Learning (ICML)*, 2020. http://proceedings.mlr.press/v119/shankar20c/shankar20c.pdf.

[80] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014. https://arxiv.org/abs/1403.6382.

[81] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[82] Yang Shu, Zhi Kou, Zhangjie Cao, Jianmin Wang, and Mingsheng Long. Zoo-tuning: Adaptive transfer from a zoo of models. In *International Conference on Machine Learning*, pages 9626–9637. PMLR, 2021.

[83] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

[84] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.

[85] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[86] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[87] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://proceedings.neurips.cc/paper/2019/file/d03a857a23b5285736c4d55e0bb067c8-Paper.pdf.

[88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[89] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[90] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1905.13549.

[91] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *TACL*, 7: 625–641, 2019.

[92] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[93] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

[94] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

[95] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[96] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo-Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. 2021. https://arxiv.org/abs/2109.01903.

[97] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 2016. https://link.springer.com/article/10.1007/s11263-014-0748-y.

[98] LI Xuhong, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pages 2825–2834. PMLR, 2018.

[99] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification, 2019. https://arxiv.org/abs/1905.00546.

[100] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. https://arxiv.org/abs/1411.1792.

[101] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[102] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021. https://arxiv.org/abs/2106.04560.

[103] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. 2017. https://arxiv.org/abs/1710.09412.

[104] Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1907.08610.

[105] Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021.

[106] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models, 2021. https://arxiv.org/abs/2109.01134.

A Additional Figures

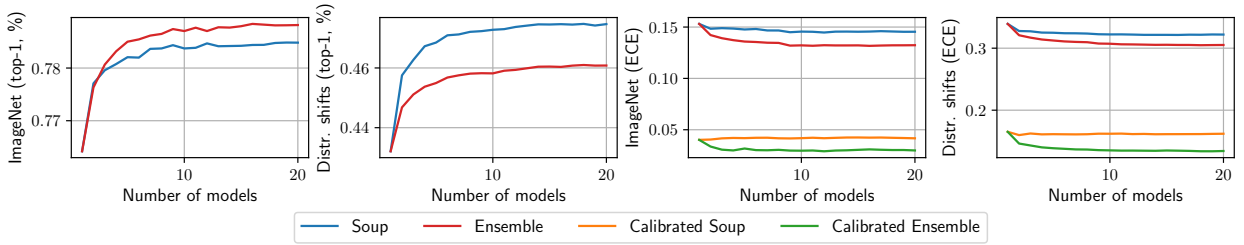

Figure A.1: Like model ensembling, model soups improve accuracy, but unlike model ensembling, model soups do not improve calibration. Expected calibration error (ECE) is computed using equal-mass binning. The soup in this figure is the uniform soup, and all models in this experiment are fine-tuned CLIP ViT-B/32 models with the same hyperparameters but different random seeds. The calibrated soup and calibrated ensemble refer to a soup and ensemble composed of models which are calibrated through temperature scaling [40]. Calibrating models before ensembling or souping had no effect on accuracy and so these curves are omitted from the plots on the left.

B Experimental Details

B.1 Learned soup

In addition to the greedy soup method described in the text, we also explore a more advanced souping procedure, which removes the sequential constraint from the greedy soup and requires only a single pass through the held out validation set. We refer to this method as the *learned soup*, as it involves learning the soup mixing coefficients for each of the ingredients on the held-out validation set. However, the learned soup has the downside of requiring all models to be simultaneously loaded in memory. In practice we combine the models on CPU before moving the parameters to GPU for each batch. For loss ℓ and validation set $\{(x_i, y_i)\}_{i=1}^n$, we find mixing coefficients $\alpha \in \mathbb{R}^k$ and temperature scaling parameter β via

$$\underset{\alpha \in \mathbb{R}^k, \beta \in \mathbb{R}}{\arg\min} \sum_{j=1}^{n} \ell\left(\beta \cdot f\left(x_j, \sum_{i=1}^{k} \alpha_i \theta_i\right), y_j\right). \tag{2}$$

In practice we find better results when α is parameterized as the output of a softmax, so that each α_i is positive and values sum to one. We optimizer the aforementioned equation with gradient based mini-batch optimization for three epochs over the held-out validation set with the AdamW otpimizer and constant learning rate 0.1.

As presented in Table 3, we also try a "by layer" variant of the learned soup. For this we learn a separate α for each layer of the network.

B.2 Error landscape visualizations

To supplement Figure 2, we provide an identical experiment but with a 10x bigger learning rate instead of 10x smaller. Results are illustrated in Figure B.1 with linear instead of log scaling for the contour lines. Since the error difference is more substantial, linear scaling was more clear. When fine-tuning with a larger learning rate, error increases on the path between the two fine-tuned solutions. All error landscape visualizations use CLIP ViT-B/32 fine-tuned on ImageNet for 10 epochs with minimal data augmentation, as used by CLIP during pre-training. When computing angles between the two fine-tuned solutions, as in Figure 3, we use the repeated weights which constitute the majority of the network parameters. We ignore gain terms which tend to skew positive if occurring before ReLU activations.

In Figure 3 we consider solutions fine-tuned with learning rates less that 10^{-4}. As in Figure B.1, if a learning rate that is large is used accuracy will decrease on the path in weight space between the two models.

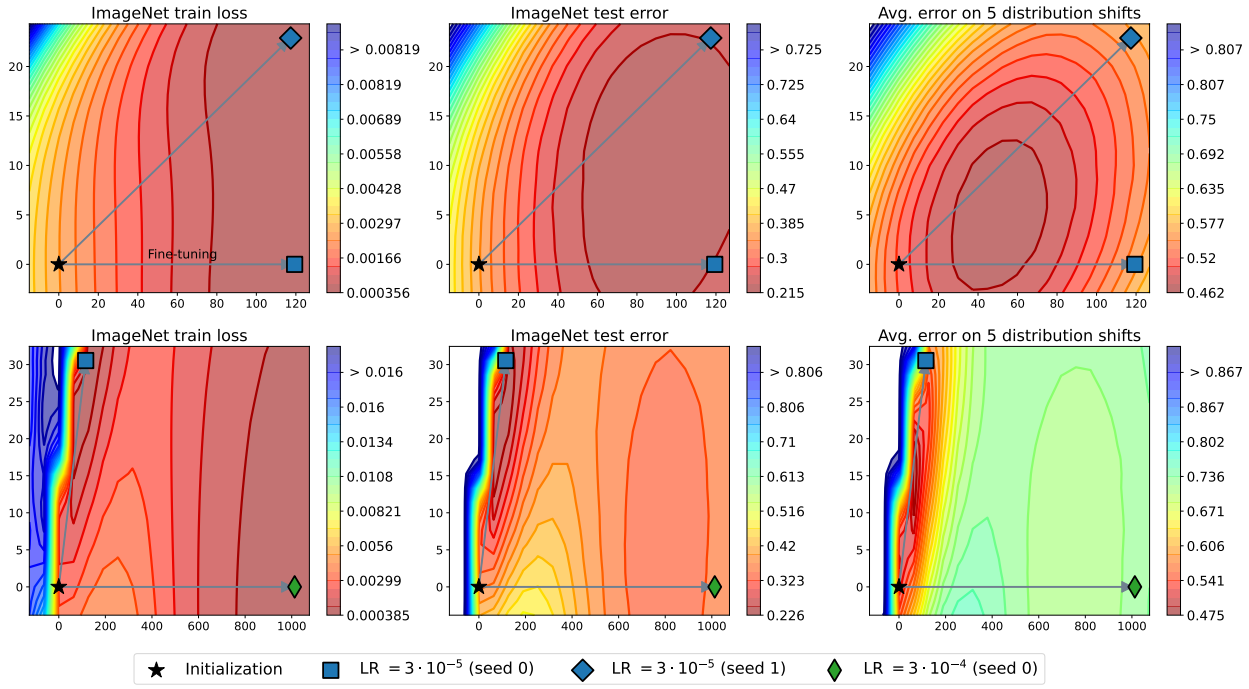

Figure B.1: Replicating Figure 2 with a 10x larger learning rate instead of 10x smaller in the second row.

B.3 Model soups

This section describes the set of hyperparameters used for searches. For all ImageNet experiments, we withhold 2% of the training set and use these examples as the held-out validation set for model selection in greedy and learned soup.

B.3.1 CLIP experiments

Unless otherwise mentioned, all experiments used the AdamW optimizer [59] with cosine annealing learning rate schedule [58] for 10 epochs at batch size 512. When necessary we discretize augmentation strength into minimal, medium, and strong. Minimal augmentation uses only a random crop consisting of 90%-100% of the total image area. Medium is the default augmentation used by the timm library [94]. Strong refers to RandAugment [20] ($N = 2$, $M = 15$).

We now provide the low level details for the hyperparemter searches, which are *standard grid*, *extreme grid*, and *random search*. The *standard grid* includes learning rates $3 \cdot 10^{-5}, 2 \cdot 10^{-5}, 1 \cdot 10^{-5}, 3 \cdot 10^{-6}$, where $2 \cdot 10^{-5}, 1 \cdot 10^{-5}$ typically perform the best. Augmentation strengths are minimal, medium, or strong. Mixup is either off or on at $\alpha = 0.5$. We consider all combinations of the above, running each hyperparameter configuration with two random seeds.

The *extreme grid* considers learning rates $3 \cdot 10^{-4}, 1 \cdot 10^{-4}, 3 \cdot 10^{-5}, 2 \cdot 10^{-5}, 1 \cdot 10^{-5}, 3 \cdot 10^{-6}, 1 \cdot 10^{-6}, 1 \cdot 10^{-7}$, where $2 \cdot 10^{-5}, 1 \cdot 10^{-5}$ typically perform the best. Augmentation strengths are minimal, medium, or strong. Mixup is either off or on at $\alpha = 0.5$. Moreover, we include the initialization in this search, which often outperforms some of the extreme learning rates but is far from the most accurate model.

The *random search* chooses learning rate $10^{-\lambda_1}$ where λ_1 is selected uniformly at random from 4 to 6. Weight decay is chosen randomly as $10^{-\lambda_2}$ where λ_2 is selected uniformly at random from 0.2 to 4. With probability 0.5, label smoothing is set to 0 and otherwise it is selected uniformly at random between 0 and 0.25. Fine-tuning epochs are chosen randomly between four and sixteen. Mixup is 0 with probability 0.5, and

otherwise is chosen uniformly at random from 0 to 0.9. With probability 1/3 we use minimal augmentation, otherwise we use randaug where M and N are chosen uniformly at random between 0 and 20 and 0 and 2 respectively.

When fine-tuning on WILDS-FMoW and WILDS-iWildCam for Figure 6, we use the same random search as when we fine-tune CLIP on ImageNet. The only difference is that we are able to use a larger ViT-L/14 model as the datasets are smaller. This also requires us to change the default batch size from 512 to 128.

B.3.2 ALIGN experiments

We fine-tuned ALIGN EfficientNet-L2 models using AdamW with weight decay of 0.1 at a resolution of 289×289 for 25 epochs, with the final layer initialized from a linear probe without data augmentation. We fine-tuned 5 models with standard Inception-style random crops (consisting of 5% to 100% of the total image area with an aspect ratio between 0.75 and 1.33) and different learning rates ($1 \cdot 10^{-6}$, $2 \cdot 10^{-6}$, $5 \cdot 10^{-6}$, $1 \cdot 10^{-5}$, and $2 \cdot 10^{-5}$). We also fine-tuned 7 additional models at a learning rate of $5 \cdot 10^{-6}$ with different data augmentation strategies. Specifically, we varied the random cropping strategy (either Inception-style crops or less aggressive crops consisting of 90% to 100% of the total image area with an aspect ratio between 0.95 and 1.05), the use of RandAugment [20] (off or $N = 2$, $M = 15$), and the use of mixup [103] (off or $\alpha = 0.5$) and trained models with all combinations of these strategies. Our soups are obtained by considering these 12 models as well as the linear probe initialization. We perform evaluation at 360×360 resolution using a square center crop from images. The accuracy we attain with greedy soup approaches that reported by Jia et al. [47], which evaluated at 600×600 resolution.

B.3.3 ViT-G/14 experiments

These models are initialized with a backbone that was pretrained on the JFT-3B dataset [102] and linear probes obtained at either the 224×224 resolution at which the ViT-G/14 was pretrained or at the 518×518 resolution used for fine-tuning. Models are fine-tuned at a batch size of 512 for either 10,000 or 20,000 steps (approximately 4 or 8 epochs) using the Adafactor optimizer [81] with learning rates of 3e-5 or 5e-5; a constant or cosine decay learning rate schedule; and softmax or binary cross-entropy loss. When fine-tuning with binary cross-entropy loss, we use a linear probe that is also trained with binary cross-entropy loss. We vary data augmentation, applying RandAugment [20], mixup [103], or CutMix [101] of varying strengths and random cropping with a minimum crop size of 5%, 70%, 90%, or 100% of the full image. When applying SAM, we consider models with perturbations either synchronized or unsynchronized across accelerators, including one model with synchronized perturbations and a combination of CutMix and SAM. All models are fine-tuned at 518×518 resolution and evaluated by rescaling test images to 550×550 (without preserving the aspect ratio) and taking a 518×518 central crop.

We manually tuned hyperparameters with the goal of maximizing single-model accuracy. After settling on the use of Adafactor as the optimizer, we included all subsequently trained models in the pool of models to be used for greedy soup. The model that performs best on the holdout set is initialized with a 224×224 linear probe and fine-tuned with a learning rate of 3e-5 and a constant learning rate decay schedule, with softmax cross-entropy loss, a minimum crop size of 90%, and CutMix with $\alpha = 0.2$. The model that performs best on the official ImageNet validation set is initialized with a 518×518 linear probe and fine-tuned at a learning rate of 3e-5 and a constant learning rate decay schedule, with softmax cross-entropy loss, a minimum crop size of 90%, CutMix with $\alpha = 0.2$, and SAM. The greedy soup contains models trained with a wide range of different hyperparameter values including different learning rates, linear probes, loss functions, and every form of data augmentation and minimum crop size investigated. Notably, although models trained with SAM with synchronized perturbations are included in the greedy soup, the greedy soup process skips over the models trained with SAM with unsynchronized perturbations because adding them produces a large drop in holdout accuracy.

B.4 Cross-dataset soups details

This section provides additional details for the findings presented in Section 3.5. When fine-tuning we initialize with CLIP ViT-B/32 and use learning rate $3 \cdot 10^{-5}$ for 10 epochs with mini-batch size of 512. We train with minimal augmentation.

C Analysis of 1D hyperparameter grids

This section asks: for a one dimensional grid of hyperparameters $\{h_a, ..., h_b\}$, how does averaging the models fine-tuned with hyperparameter configurations h_a and h_b corresponding to the endpoints compare with picking the best individual model fine-tuned with hyperparameter configuration $h \in \{h_a, ..., h_b\}$?

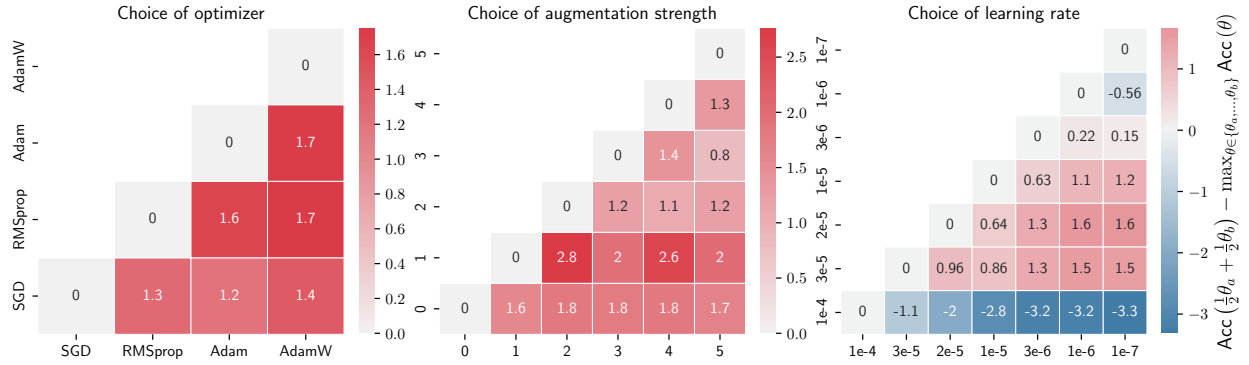

Figure C.1: Analysis of 1D hyperparameter grids, where the average of models at the endpoints often outperforms the best individual model in the grid. In particular, colors and numbers indicate the percentage point improvement obtained by averaging the models on the x and y axis versus taking the best individual model in the range between them. Results are shown for the CLIP ViT-B/32 model fine-tuned on ImagNet.

The results are illustrated in Figure C.1, where each square represents a grid $\{h_a, ..., h_b\}$. The average of the endpoints often outperforms the best individual model in the grid. A notable exception is when the learning rate 10^{-4} is the left endpoint of the grid. As this experiment uses AdamW, this learning rate is too high for fine-tuning and, unlike the examples in Figure 2, there is a high error barrier between the two fine-tuned solutions (see Figure B.1, lower right for example).

When varying optimizer we use minimal data augmentation and LR $3 \cdot 10^{-5}$ for RMSProp [86], Adam [48], and AdamW [59]. SGD requires a larger learning rate, and so we use 0.1. When varying augmentation strength, we use minimal data augmentation and LR $3 \cdot 10^{-5}$.

D Additional grid searches and initializations

This section recreates Figure 5 with different initializations (linear probe initialization or zero-shot) and different grid searches (standard and extreme grid) when fine-tuning CLIP ViT-B/32. The standard and extreme grid searches are described in Section B.3.1.

Figure D.1 considers the linear probe (LP) initialization and the *standard grid*. Figure D.2 considers the linear probe (LP) initialization and the *extreme grid*. Figure D.3 considers the zero-shot initialization and the *standard grid*. Figure D.4 considers the zero-shot initialization and the *extreme grid*.

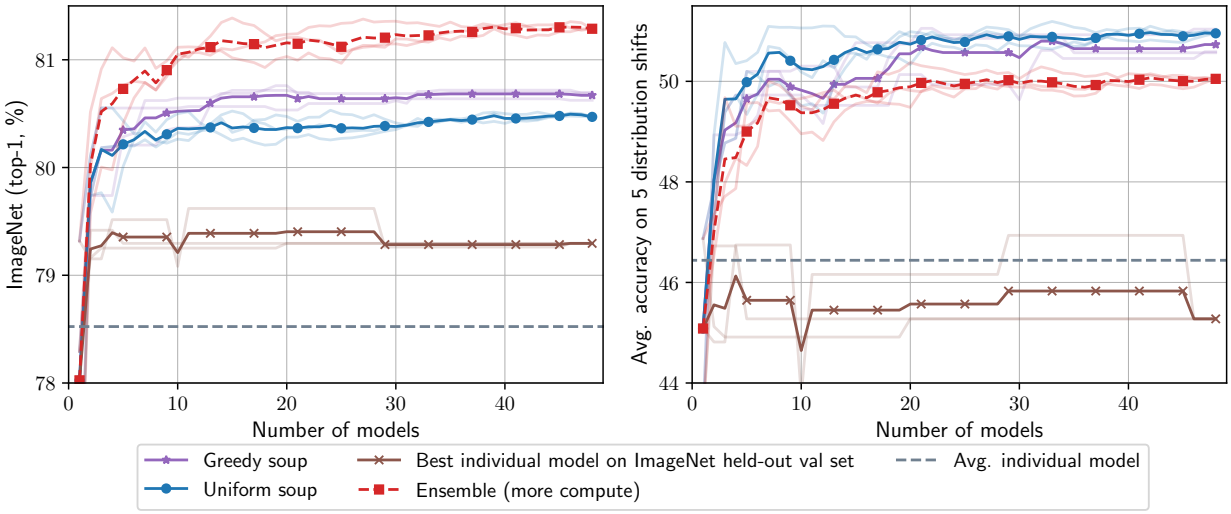

Figure D.1: Replicating Figure 5 with the LP initialization and the *standard grid* hyperparameter search.

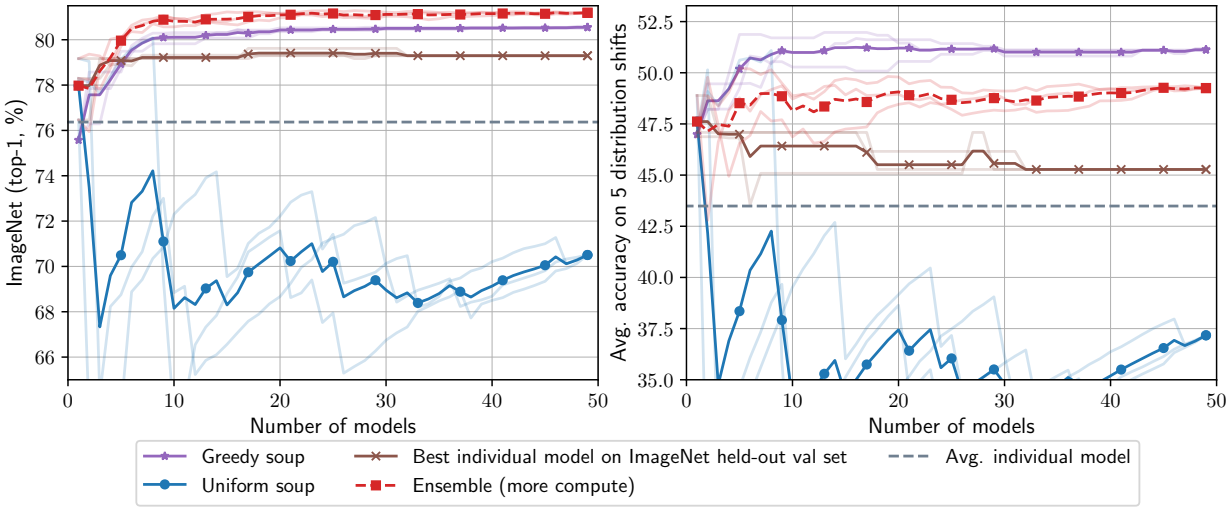

Figure D.2: Replicating Figure 5 with the LP initialization and the *extreme grid* hyperparameter search.

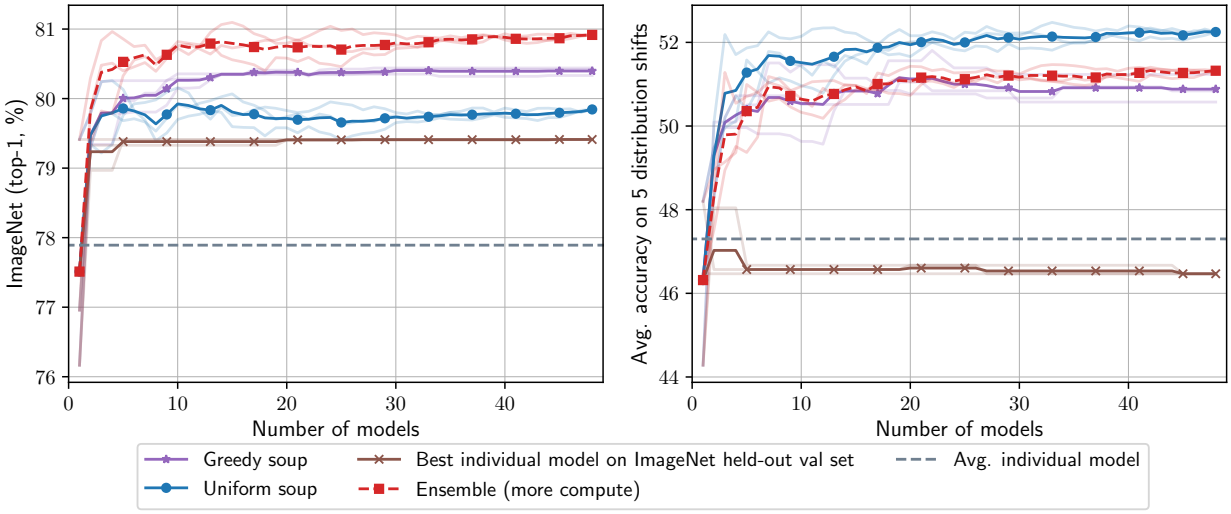

Figure D.3: Replicating Figure 5 with the zero-shot initialization and the *standard grid* hyperparameter search.

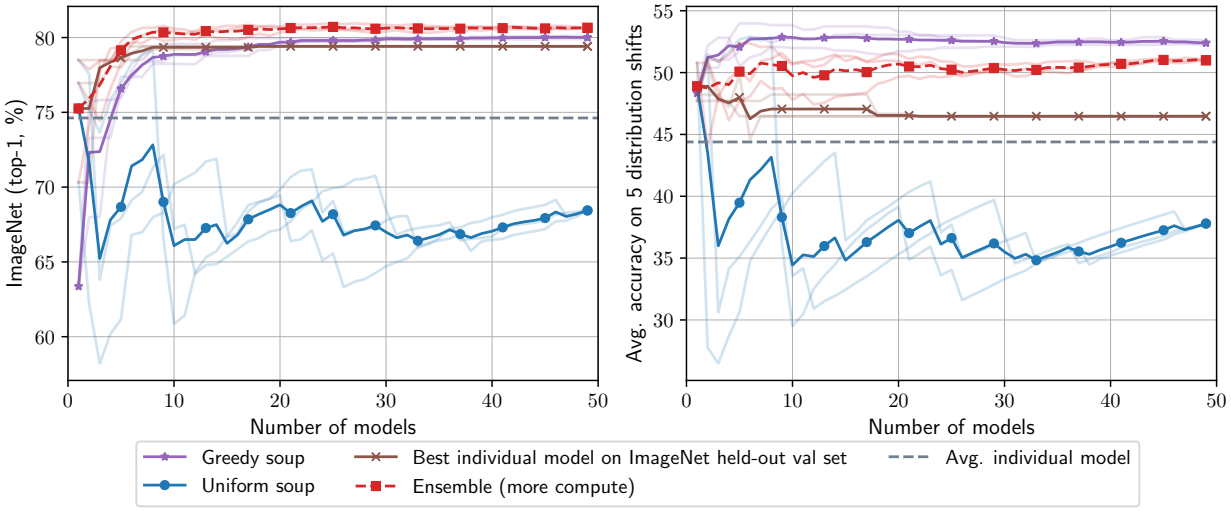

Figure D.4: Replicating Figure 5 with the zero-shot initialization and the *extreme grid* hyperparameter search.

E Additional fine-tuning and pre-training datasets

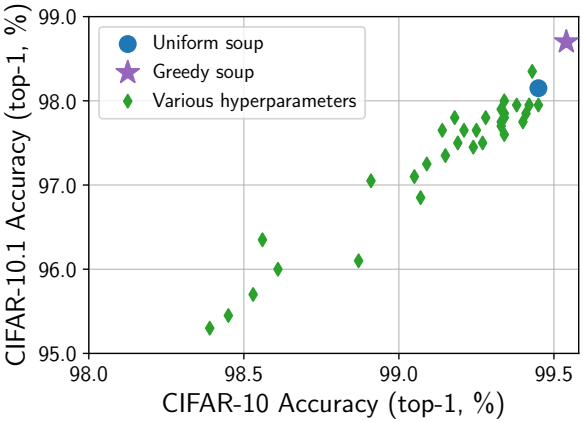

Figure E.1: Fine-tuning a CLIP ViT-L model on CIFAR-10 [53] with the random hyperparameter search described in Section B.3.1. The y-axis displays accuracy on CIFAR-10.1 [75], a reproduction of CIFAR-10 with a distribution shift.

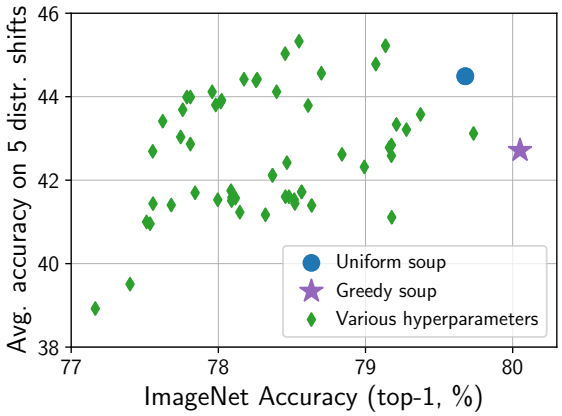

Figure E.2: Fine-tuning on ImageNet, using a ViT-B/32 [30] pre-trained on ImageNet-22k [23].

In this section we explore fine-tuning or pre-training on additional datasets. Figure E.1 displays results for fine-tuning a CLIP ViT-L model on CIFAR-10 [53]. The y-axis of Figure E.1 displays accuracy on CIFAR-10.1 [75], a reproduction of CIFAR-10 with a distribution shift. The individual models are fine-tuned with the random hyperparameter search described in Section B.3.1.

Next, Figure E.2 shows results when fine-tuning a ViT-B/32 [30] model pre-trained on ImageNet-22k [23] and fine-tuned on ImageNet. This differs from many of our other experiments as the dataset used for pre-training is smaller and less diverse. While the greedy soup offers an improvement, the improvement is less substantial than, e.g., Figure 1 which uses the same model and hyperparameter search but a different pre-training dataset.

Finally, we fine-tune a ViT-B/32 model five times on ImageNet, using the best hyperparameters found by the hyperparameter sweep, varying only the random seed. This experiment is conducted both for a model pre-trained on ImageNet-22k [23] and a pre-trained CLIP model. The results are shown in Figure E.3, comparing, for an experimental budget of $1 \leq k \leq 5$ models: (i) the individual model with random seed k, (ii) the model soup composed of models with random seeds 1 through k, and (iii) the ensemble composed of models with random seeds 1 through k. The performance of the model soup appears correlated with the performance of the ensemble, as we find that CLIP models are more amenable to both ensembling and souping than models pre-trained on ImageNet-22k.

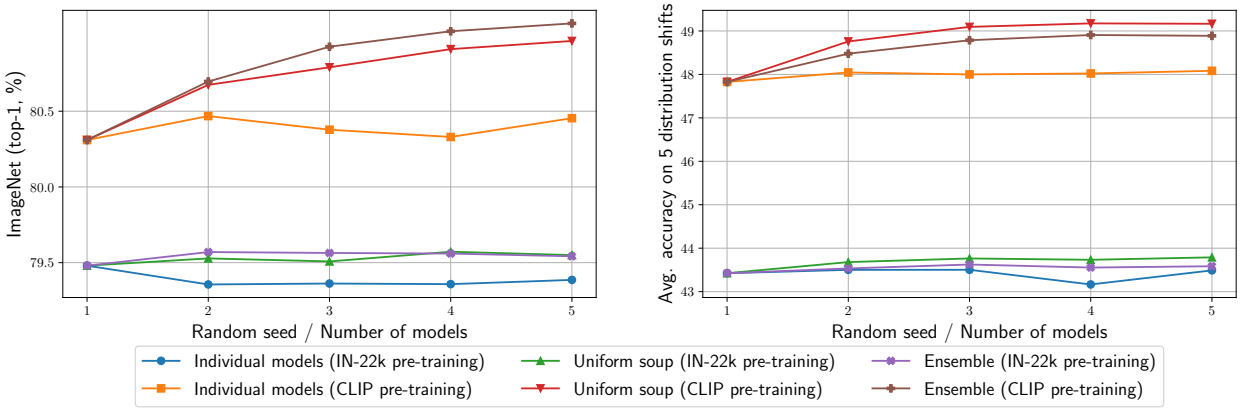

Figure E.3: For a CLIP and ImageNet-22k pre-trained ViT-B/32 model, we use the best hyperparameters found by the hyperparameter sweep to fine-tune multiple times, varying only the random seed. For an experimental budget of $1 \leq k \leq 5$ models, we show: (i) the individual model with random seed k, (ii) the model soup composed of models with random seeds 1 through k, and (iii) the ensemble composed of models with random seeds 1 through k.

F Analytical comparison details

F.1 Notation and preliminaries

We begin by restating and adding to the notation used in Section 4. For a model with parameter vector $\theta \in \mathbb{R}^d$ and input vector x, we let $f(x;\theta) \in \mathbb{R}^C$ denote the model's logit output for C-way classification. Throughout, we fix two endpoint models θ_0 and θ_1, and for an interpolation parameter $\alpha \in [0,1]$ define

$$\theta_\alpha := (1-\alpha)\theta_0 + \alpha\theta_1, \quad \text{and} \quad f_\alpha^{\text{soup}}(x) := f(x;\theta_\alpha)$$

to be the "soup" weight averaged model and its corresponding logits. We also write

$$f_\alpha^{\text{ens}}(x) := (1-\alpha)f(x;\theta_0) + \alpha f(x;\theta_1)$$

for the logits of the ensemble model. We write

$$\delta = \theta_1 - \theta_0$$

for the difference of the two endpoints.

For a logit vector $f \in \mathbb{R}^C$ and a ground-truth label y, denote the cross-entropy loss by

$$\ell(f;y) = \log\left(\sum_{y'} \exp\{f_{y'} - f_y\}\right).$$

For some distribution over x, y we write the expected β-calibrated log losses of the soup and ensemble as

$$\mathcal{L}_\alpha^{\text{soup}} = \mathbb{E}_{x,y}\ell(\beta f(x;\theta_\alpha), y) \quad \text{and} \quad \mathcal{L}_\alpha^{\text{ens}} = \mathbb{E}_{x,y}\ell(\beta f_\alpha^{\text{ens}}(x), y),$$

respectively.

We have the following expression for the derivatives of cross entropy w.r.t. logits. The gradient is

$$\nabla_f \ell\,(f,y) = p_{\text{sftmx}}(f) - e^{(y)},$$

where $e^{(i)}$ is the ith standard basis vector and $p_{\text{sftmx}}(f) \in \mathbb{R}^C$ has $e^{f_i}/\sum_j e^{f_j}$ in its ith entry. The Hessian is

$$\nabla_f^2 \ell\left(f, y\right) = \operatorname{diag}\left(p_{\text{sftmx}}\left(f\right)\right) - [p_{\text{sftmx}}(f)][p_{\text{sftmx}}(f)]^T,$$

so that for any $v \in \mathbb{R}^C$, we have

$$v^T \nabla_f^2 \ell\left(f, y\right) v = \operatorname{Var}_{Y \sim p_{\text{sftmx}}(f)}[v_Y].$$

Finally, we use $\delta^T \nabla f(x;\theta)$ to denote a vector in $\mathbb{R}^C$ whose ith entry is $\delta^T \nabla [f(x;\theta)]_i$. Similarly, $\delta^T \nabla^2 f(x;\theta)\delta$ denotes a vector in $\mathbb{R}^C$ whose ith entry is $\delta^T [\nabla^2 f(x;\theta)]_i \delta$, where gradients and Hessian are with respect to θ.

F.2 An exact expression for logit difference

We use the fundamental theorem of calculus and elemntary algebraic manipulation to obtain an exact integral form for the difference between the soup and ensemble logits. To streamline notation we drop the dependence of the logits on the input x.

$$\begin{aligned}
f_\alpha^{\text{ens}} - f_\alpha^{\text{soup}} &= (1-\alpha)\left[f\left(\theta_0\right) - f\left(\theta_\alpha\right)\right] + \alpha\left[f\left(\theta_1\right) - f\left(\theta_\alpha\right)\right] \\
&= -\left(1-\alpha\right)\int_0^\alpha \delta^T \nabla f\left(\theta_t\right) \mathrm{d}t + \alpha \int_\alpha^1 \delta^T \nabla f\left(\theta_t\right) \mathrm{d}t \\
&= -\left(1-\alpha\right)\int_0^\alpha \left[\delta^T \nabla f\left(\theta_\alpha\right) + \int_\alpha^t \delta^T \nabla f\left(\theta_\tau\right) \delta \mathrm{d}\tau\right] \mathrm{d}t + \alpha \int_\alpha^1 \left[\delta^T \nabla f\left(\theta_\alpha\right) + \int_\alpha^t \delta^T \nabla f\left(\theta_\tau\right) \delta \mathrm{d}\tau\right] \mathrm{d}t \\
&= -\left(1-\alpha\right)\int_0^\alpha \int_\alpha^t \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) \mathrm{d}\tau \mathrm{d}t + \alpha \int_\alpha^1 \int_\alpha^t \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) \mathrm{d}\tau \mathrm{d}t \\
&= \left(1-\alpha\right)\int_0^\alpha \int_t^\alpha \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) \mathrm{d}\tau \mathrm{d}t + \alpha \int_\alpha^1 \int_\alpha^t \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) \mathrm{d}\tau \mathrm{d}t \\
&= \left(1-\alpha\right)\int_0^\alpha \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) \mathrm{d}\tau \int_0^\tau \mathrm{d}t + \alpha \int_\alpha^1 \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) \mathrm{d}\tau \int_\tau^1 \mathrm{d}t \\
&= \int_0^1 \left(\delta^T \nabla^2 f\left(\theta_\tau\right) \delta\right) w_\alpha\left(\tau\right) \mathrm{d}\tau, \qquad (3)
\end{aligned}$$

where

$$w_\alpha\left(\tau\right) = \begin{cases} \left(1-\alpha\right)\tau & \tau \le \alpha \\ \alpha\left(1-\tau\right) & \text{otherwise} \end{cases} = \min\left\{\left(1-\alpha\right)\tau, \alpha\left(1-\tau\right)\right\}.$$

Note that

$$\int_0^1 w_\alpha(\tau)\mathrm{d}\tau = \frac{\alpha(1-\alpha)}{2}.$$

F.3 Derivation of approximation

We continue to suppress the dependence on x in order to simplify notation. We begin with the following first order approximation of the pointwise log-loss difference between the ensemble and soup, which is also a lower bound due to convexity.

$$\ell(f_\alpha^{\text{ens}}; y) - \ell(f_\alpha^{\text{soup}}; y) \approx [\nabla_f \ell(f_\alpha^{\text{ens}}; y)]^T (f_\alpha^{\text{ens}} - f_\alpha^{\text{soup}}) + O\left(((f_\alpha^{\text{ens}} - f_\alpha^{\text{soup}})^2)\right).$$

Now, we approximate the ensemble and soup logit difference using eq. 3 by assuming that $\delta^T \nabla^2 f\left(\theta_\tau\right) \delta \approx \delta^T \nabla^2 f\left(\theta_\alpha\right) \delta$ for all $\tau \in [0,1]$; this holds when the logits are approximately quadratic along the line between

the checkpoints. The resulting approximation is

$$f_\alpha^{\text{ens}} - f_\alpha^{\text{soup}} \approx c_\alpha \, \delta^T \nabla^2 f\left(\theta_\alpha\right) \delta + O\left(\max_{\tau \in [0,1]} |\nabla^3 f(\theta_\tau)[\delta^{\otimes 3}]|\right), \quad \text{where } c_\alpha \coloneqq \frac{(1-\alpha)\alpha}{2}.$$

Combining the two approximation above, we obtain

$$\ell(f_\alpha^{\text{ens}}; y) - \ell(f_\alpha^{\text{soup}}; y) \approx c_\alpha \, [\nabla_f \ell(f_\alpha^{\text{ens}}; y)]^T \delta^T \nabla^2 f\left(\theta_\alpha\right) \delta.$$

To relate this expression to the Hessian of the loss with respect to the parameters, we note that for any θ (by the chain rule)

$$\delta^T \nabla_\theta^2 \ell(f(\theta); y) \delta = [\delta^T \nabla f(\theta)]^T \nabla_f^2 \ell(f(\theta); y) [\delta^T \nabla f(\theta)] + \nabla_f \ell(f(\theta); y)]^T \delta^T \nabla^2 f(\theta) \delta.$$

When setting $\theta = \theta_\alpha$, we note that the second term on the RHS is (up to a constant) our approximation for the loss difference). Recalling the expression for the cross-entropy Hessian, the first term is

$$[\delta^T \nabla f(\theta)]^T \nabla_f^2 \ell(f(\theta); y) [\delta^T \nabla f(\theta)] = \text{Var}_{Y \sim p_{\text{sftmx}}(f(\theta))} \left[\delta^T \nabla f(\theta)\right].$$

As a final approximation, we let

$$\delta^T \nabla f(\theta_\alpha) \approx f(\theta_1) - f(\theta_0) + O\left(\delta^T \nabla^2 f(\theta) \delta\right);$$

this holds when logits are too far from linear in θ.

Substituting back and making x explicit, we obtain

$$\ell(f_\alpha^{\text{soup}}(x); y) - \ell(f_\alpha^{\text{ens}}(x); y) \approx -c_\alpha \frac{\mathrm{d}^2}{\mathrm{d}\alpha^2} \ell(f_\alpha^{\text{soup}}(x); y) + c_\alpha \, \text{Var}_{Y \sim p_{\text{sftmx}}(f_\alpha^{\text{soup}}(x))} \left[f(x; \theta_1) - f(x; \theta_0)\right],$$

where we have used

$$\delta^T \nabla_\theta^2 \ell(f_\alpha^{\text{soup}}(x); y) \delta = \frac{\mathrm{d}^2}{\mathrm{d}\alpha^2} \ell(f_\alpha^{\text{soup}}(x); y).$$

Scaling all logits by β, the approximation becomes

$$\ell(\beta f_\alpha^{\text{soup}}(x); y) - \ell(\beta f_\alpha^{\text{ens}}(x); y) \approx -c_\alpha \frac{\mathrm{d}^2}{\mathrm{d}\alpha^2} \ell(\beta f_\alpha^{\text{soup}}(x); y) + c_\alpha \, \beta^2 \text{Var}_{Y \sim p_{\text{sftmx}}(\beta f_\alpha^{\text{soup}}(x))} \left[f(x; \theta_1) - f(x; \theta_0)\right].$$

Averaging the result over x, we arrive at the approximation (1), which we repeat here for ease of reference:

$$\mathcal{L}_\alpha^{\text{soup}} - \mathcal{L}_\alpha^{\text{ens}} \approx -c_\alpha \frac{\mathrm{d}^2}{\mathrm{d}\alpha^2} \mathcal{L}_\alpha^{\text{soup}} + c_\alpha \, \beta^2 \mathbb{E}_x \text{Var}_{Y \sim p_{\text{sftmx}}(\beta f_\alpha^{\text{soup}}(x))} \left[f(x; \theta_1) - f(x; \theta_0)\right].$$

F.4 Detailed empirical evaluations

Evaluation setup. We evaluated our bounds on checkpoints from the ViT-B/32 fine-tuning experiments from the *extreme grid* search described in Section B.3.1. We selected three learning rate values (10^{-6}, 10^{-5} and 10^{-4}), two levels augmentation (none and RandAugment+MixUp), and considered two different random seeds (0 and 1). From these checkpoints (as well as the initialization) we constructed the following (θ_0, θ_1) pairs:

- All pairs with different learning rate, the same augmentation level and seed 0,
- All pairs with the same learning rate, different augmentation level and seed 0,
- All pairs with the same learning rate and augmentation level, but different seeds,
- All checkpoints with seed 0 coupled with the initialization.

This results in 21 pairs overall. For each pair and each $\alpha \in \{0, 0.1, \ldots, 0.9, 1.0\}$ we evaluated err_α, $\text{err}_\alpha^{\text{ens}}$, $\mathcal{L}_\alpha^{\text{soup}}$, $\mathcal{L}_\alpha^{\text{ens}}$, as well as the approximation (1). We performed this evaluation on the ImageNet validation set as well as on the 5 distribution shift test sets considered throughout this paper.

The effect of temperature calibration. Since our ultimate goal is to accurately predict the difference in error rather than the difference in loss, we introduce the inverse-temperature parameter β to the loss, and tune it to calibrate the soup model. Specifically, for every model pair, value of α and test set, we take $\beta = \arg\min_\beta \mathbb{E}_{x,y}\ell(\beta f_\alpha^{\text{soup}}(x); y)$.

While choosing β based on the soup rather the ensemble might skew the loss in favor of the soup, it has no effect on the difference in prediction error. Moreover, in preliminary experiments calibrating the ensemble produced very similar results. In contrast, as shown in Figure 9, fixing $\beta = 1$ throughout results in far poorer prediction of the difference in error.

G Additional baselines

This section explores additional baselines for model soups, including distillation from an ensemble as in Hinton et al. [44] (Table G.1), fix-augmentation as in Touvron et al. [87] (Table G.2), weight-averaging along a trajectory as in Szegedy et al. [85] (Table G.3), and Sharpness Aware Minimization as in Foret et al. [31] (Table G.4).

Unless otherwise mentioned, we fine-tune CLIP ViT-B/32 models with AdamW [59] and cosine annealing learning rate [58] for 10 epochs on ImageNet with a learning rate of 2e-5 and medium augmentation (data augmentation policies are discussed in more detail in Section B.3.1).

We explore the baseline of distillation [44, 45] from the ensemble of three models trained with different data augmentation. As previously reported [4, 11], we find that it improves accuracy to run distillation with data augmentation. Unfortunately, this substantially increases the computational resources necessary to distill from the ensemble. As we cannot cache the predictions of the models in the ensemble, it is necessary to perform a forward pass for each model in the ensemble at each step of fine-tuning. This makes distilling from an ensemble similarly expensive as training the models which constitute the ensemble. Nevertheless, as illustrated in Table G.1, model soups still perform favorably.

Table G.1 also introduces stochastic augmentation. For each data point, stochastic augmentation randomly applies minimal, medium, or strong data augmentation. Additionally, Table G.2 explores an alternative method for merging augmentations together. This augmentation policy, which we refer to as fix-aug, is introduced by Touvron et al. [87]. For fix-aug, strong augmentation is used for all but the final epoch, which uses minimal augmentation.

Next, Table G.3 applies model soups to solutions which already average along the fine-tuning trajectory. Methods for averaging along an individual optimization trajectory include exponential moving averages (EMA) [85] and stochastic weight averages (SWA) [46]. Since accuracy is high even from initialization we use EMA. While EMA improves the accuracy of a single model, we find that models without EMA are more amenable to souping. Regardless, the model soup improves over the best single model with EMA.

Finally, Table G.4 explores the relation between model soups and sharpness-aware minimization (SAM) [31]. In line with previous results, we find that SAM improves accuracy over vanilla fine-tuning. Souping two models trained with SAM improves over either individual model, although the magnitude of the gain is smaller than for vanilla fine-tuning. Souping models trained with and without SAM yields higher accuracy than souping models trained only with vanilla fine-tuning or only with SAM.

Table G.1: Comparing model soups to network distillation from an ensemble of models trained with different data augmentations. Stochastic data augmentation randomly applies minimal, medium, or strong data augmentation.

	ImageNet	Distribution shifts
Individual model (LR 3e-05, minimal aug)	76.42	43.21
Individual model (LR 3e-05, medium aug)	78.83	43.55
Individual model (LR 3e-05, strong aug)	79.08	43.75
Individual model (LR 3e-05, stochastic aug)	78.94	45.04
Individual model (LR 3e-05, stochastic aug 3x epochs)	78.38	42.18
Distillation from the ensemble (LR 3e-05, no aug)	78.59	43.45
Distillation from the ensemble (LR 3e-05, stochastic aug)	79.79	45.63
Soup minimal, medium, and strong aug (LR 3e-05)	80.24	47.97
Ensemble minimal, medium, and strong aug (LR 3e-05)	80.19	46.33
Individual model (LR 1e-05, minimal aug)	77.19	47.98
Individual model (LR 1e-05, medium aug)	79.51	46.74
Individual model (LR 1e-05, strong aug)	79.33	46.62
Individual model (LR 1e-05, stochastic aug)	79.48	48.07
Individual model (LR 1e-05, stochastic aug 3x epochs)	79.59	46.89
Distillation from the ensemble (LR 1e-05, no aug)	79.13	47.28
Distillation from the ensemble (LR 1e-05, stochastic aug)	79.88	47.49
Soup minimal, medium, and strong aug (LR 1e-05)	80.08	49.75
Ensemble minimal, medium, and strong aug (LR 1e-05)	80.17	49.36

Table G.2: Comparing models soups of different augmentations with another method which combines different augmentation strategies—fix aug, as described in Touvron et al. [87]. For fix aug we use strong data augmentation for all except the final epoch for which we apply minimal aug.

	ImageNet	Distribution shifts
Individual model (LR 3e-05, minimal aug)	76.42	43.21
Individual model (LR 3e-05, medium aug)	78.83	43.55
Individual model (LR 3e-05, strong aug)	79.08	43.75
Individual model (LR 3e-05, fix aug)	79.43	45.46
Individual model (LR 3e-05, fix aug 4x epochs)	78.57	41.53
Soup minimal, medium, and strong aug (LR 3e-05)	80.24	47.97
Soup minimal, medium, strong, and fix aug (LR 3e-05)	80.41	48.14
Individual model (LR 1e-05, minimal aug)	77.19	47.98
Individual model (LR 1e-05, medium aug)	79.51	46.74
Individual model (LR 1e-05, strong aug)	79.33	46.62
Individual model (LR 1e-05, fix aug)	79.70	48.18
Individual model (LR 1e-05, fix aug 4x epochs)	79.96	45.86
Soup minimal, medium, and strong aug (LR 1e-05)	80.08	49.75
Soup minimal, medium, strong, and fix aug (LR 1e-05)	80.17	49.71

Table G.3: Applying model soups to methods which average models along the trajectory. While taking the exponential moving average (EMA) of weights along the trajectory can improve the performance of a single model, the EMA solution is less amenable to souping.

	ImageNet	Distribution shifts
Individual model (no EMA, LR 3e-05, minimal aug)	76.42	43.21
Individual model (no EMA, LR 3e-05, medium aug)	78.83	43.55
Individual model (no EMA, LR 3e-05, strong aug)	79.08	43.75
Soup minimal, medium, and strong aug without EMA (LR 3e-05)	80.24	47.97
Individual model (EMA decay 0.9999, LR 3e-05, minimal aug)	77.61	47.45
Individual model (EMA decay 0.9999, LR 3e-05, medium aug)	79.37	46.89
Individual model (EMA decay 0.9999, LR 3e-05, strong aug)	79.17	46.85
Soup minimal, medium, strong aug with EMA (LR 3e-05)	79.76	49.69
Individual model (no EMA, LR 1e-05, minimal aug)	77.19	47.98
Individual model (no EMA, LR 1e-05, medium aug)	79.51	46.74
Individual model (no EMA, LR 1e-05, strong aug)	79.33	46.62
soup minimal, medium, strong without EMA (LR 1e-05)	80.08	49.75
Individual model (EMA decay 0.9999, LR 1e-05, minimal aug)	77.47	50.94
Individual model (EMA decay 0.9999, LR 1e-05, medium aug)	78.93	49.76
Individual model (EMA decay 0.9999, LR 1e-05, strong aug)	78.85	49.02
Soup minimal, medium, and strong aug with EMA (LR 1e-05)	79.16	51.76

Table G.4: Applying model soups to models trained with sharpness aware minimization (SAM) [31].

	ImageNet	Distribution shifts
Vanilla fine-tuning (seed 0)	79.32	45.09
Vanilla fine-tuning (seed 1)	79.16	45.12
SAM fine-tuning (seed 0)	79.61	43.78
SAM fine-tuning (seed 1)	79.59	43.79
Soup (vanilla fine-tuning, seeds 0 and 1)	79.78	46.46
Soup (SAM fine-tuning, seeds 0 and 1)	79.85	44.44
Soup (vanilla fine-tuning and SAM fine-tuning, seed 0)	80.04	45.38

H Text classification datasets

We study four text classification datasets from the GLUE benchmark [89].

Microsoft Research Paraphrase Corpus (MRPC; [28]) contains pairs of sentences, labeled as either nearly semantically equivalent, or not. The dataset is evaluated using the average of F_1 and accuracy. The training set consists of 3.7 thousand samples and the validation set of 409 samples.

Recognizing Textual Entailment (RTE; [89]) contains pair of sentences, and the task is to predict whether the first sentence (the premise) entails or contradicts the second sentence (the hypothesis). The data is originally from a series of datasets [21, 5, 37, 9]. The dataset is evaluated using classification accuracy. The training set consists of 2.5 thousand samples and the validation set of 277 samples.

Corpus of Linguistic Acceptability (CoLA; [91]) contains sentences labeled as either grammatical or ungrammatical. Models are evaluated on Matthews correlation (MCC; [63]), which ranges between -1 and 1. The training set consists of 8.6 thousand samples and the validation set consists of 1043 samples.

Stanford Sentiment Treebank (SST-2; [84]) contains sentences labelled as expressing *positive* or *negative* sentiment, collected from movie reviews. The dataset is evaluated using classification accuracy. The training set consists of 67 thousand samples and the validation set consists of 873 samples.

I Fine-tuning details for text classification tasks

Each model is fine-tuned 32 times on each dataset, performing a random hyperparameter search. The learning rate is chosen uniformly in log space over $[10^{-6}, 10^{-3}]$, the batch size is chosen uniformly from $\{8, 16, 32, 64\}$ and the number of epochs from $\{2, 3, 5\}$. Evaluation is conducted once at the end of training, without early stopping. We use a maximum sequence length of 128 tokens and train with Adam [48] using $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, gradient clipping of 1.0, no weight decay, and with the learning rate being decayed linearly to zero at the end of training. We use pre-trained weights from the Huggingface Transformers library [95]. For BERT models, we use the uncased version.

Fine-tuning occurs without any additional parameters to avoid distorting the features from the pre-trained models [54]. For such, the classification tasks are adapted to be suited to the pre-training objective of BERT and T5. For T5, the tasks are cast as a sequence-to-sequence problem. For instance, for sentiment analyses, an example is to predict "`A) positive`" from "`sentence: The best movie I've ever seen! | options: A) positive B) negative | label:`". For BERT, the tasks are cast as a masked language modeling problem. For instance, for linguistic acceptability, an example is to predict "`A) acceptable`" for the inputs "`sentence: model soups are grammatical. | options: A) acceptable B) unacceptable | label: [MASK] [MASK] [MASK]`". For evaluation, we select which of the options is given the highest probability according to the model.

The full set of results is shown in Table I.1. On 10 out of the 20 combinations of models and datasets, the greedy soup shows better performance than the best individual model from the hyperparameter search. Uniform soups show worse performance than the best individual model on all experiments, which could be an artifact of the broad range of hyperparameters used in the search. While the experiments varied only basic hyperparameters such as learning rate and batch size, we hypothesize that a broader set of hyperparameter choices (e.g. data augmentation [92, 60]) could lead to more diverse models and better soups.

Finally, as a word of caution for practitioners, we remind readers that many recent language models have tied weights on the output and embedding layers [71]. For this reason, caution is needed when averaging models in-place; doing so might inadvertently lead to undesired behavior.

Table I.1: Performance of model soups on four text classification datasets from the GLUE benchmark [89].

Model	Method	MRPC	RTE	CoLA	SST-2
BERT-base [25]	Best individual model	88.3	61.0	59.1	92.5
	Uniform soup	76.0	52.7	0.0	89.9
	Greedy soup	88.3	61.7	59.1	93.0
BERT-large [25]	Best individual model	88.8	56.7	**63.1**	92.2
	Uniform soup	15.8	52.7	1.90	50.8
	Greedy soup	88.8	56.7	**63.1**	92.3
T5-small [74]	Best individual model	89.7	70.0	42.2	91.7
	Uniform soup	82.7	61.7	10.4	91.1
	Greedy soup	89.7	70.0	43.0	91.7
T5-base [74]	Best individual model	91.8	78.3	58.8	94.6
	Uniform soup	86.4	71.8	12.3	94.6
	Greedy soup	92.4	79.1	60.2	94.7
T5-large [74]	Best individual model	**93.4**	82.7	61.7	**96.3**
	Uniform soup	74.8	50.2	0.00	96.0
	Greedy soup	**93.4**	**84.8**	62.7	**96.3**