Model soups: averaging weights of multiple fine-tuned models

Alexander Gaponov

Motivation

- High accuracy on both in/out distribution samples
- Fast inference (killer-feature)
- Easy to implement

Method description

- One pre-trained model $f(x, \theta)$
- Fine-tuning multiple models θ_i = FineTune(θ_0, h_i), h_i is hyperparameters and augmentations configuration
- Averaging received weights to build model soup

Ways of weights averaging. Uniform soup

- Uniform soup
- Greedy soup
- Learned soup

	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 I	$\mathcal{O}(1)$

Greedy soup

Recipe 1 GreedySoup

Input: Potential soup ingredients $\{\theta_1, ..., \theta_k\}$ (sorted in decreasing order of $\mathsf{ValAcc}(\theta_i)$).

$\mathsf{ingredients} \leftarrow \{\}$

for $i = 1$ **to** k **do**

  if $\mathsf{ValAcc}(\mathsf{average}(\mathsf{ingredients} \cup \{\theta_i\})) \geq \mathsf{ValAcc}(\mathsf{average}(\mathsf{ingredients}))$ **then**

    $\mathsf{ingredients} \leftarrow \mathsf{ingredients} \cup \{\theta_i\}$

return $\mathsf{average}(\mathsf{ingredients})$

Learned soup

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

Experiments setup

- Training CLIP, ALIGN, BASIC, ViT-G/14
- Testing models on ImageNet and it’s 5 distribution shifts
- Training BeRT, T5
- Testing on 4 classification tasks from the GLUE benchmark
- Choosing hyperparameters randomly

Soups improve accuracy of ALIGN

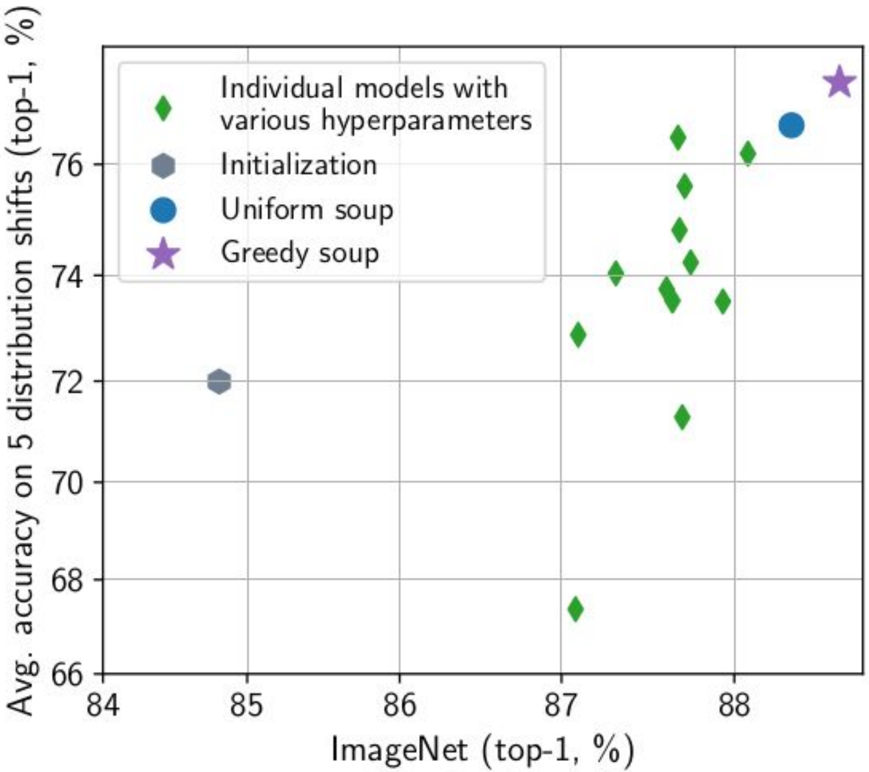


Figure 5: *Model soups* improve accuracy when fine-tuning ALIGN.

Soups improve accuracy of CLIP

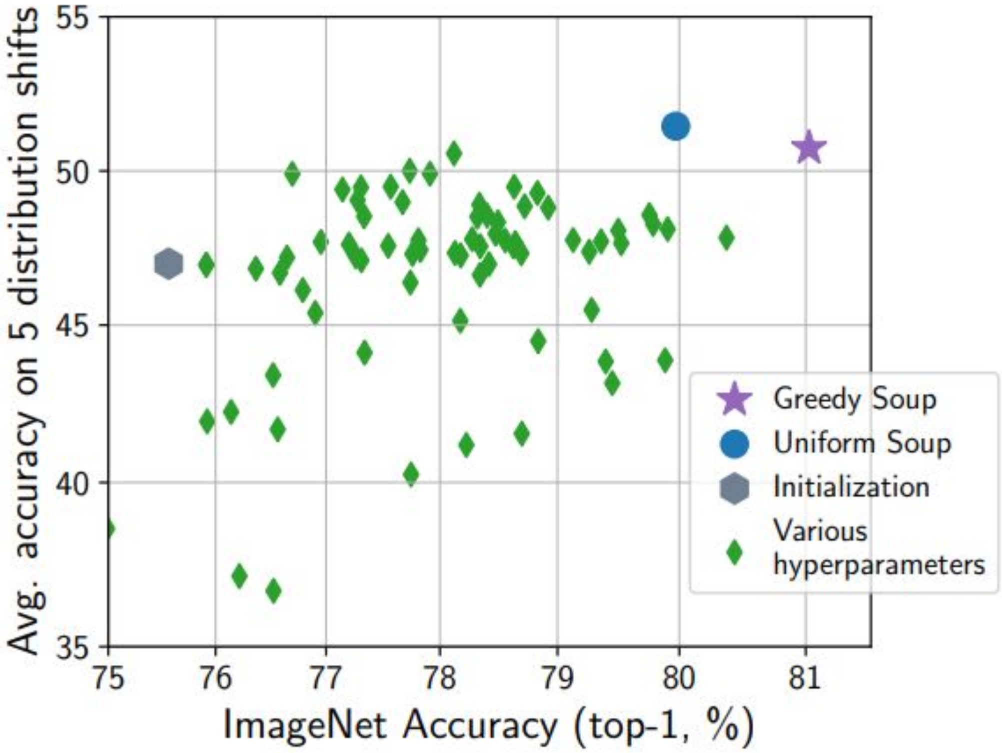


Figure 1: *Model soups* improve accuracy over the best individual model when performing a large, random hyperparameter search for fine-tuning a CLIP ViT-B/32 model on ImageNet.

Fine-tuning CLIP ViT-B/32

	ImageNet	Dist. shifts
Best individual model	80.38	47.83
Second best model	79.89	43.87
Uniform soup	79.97	51.45
Greedy soup	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

Accuracy depending on number of models

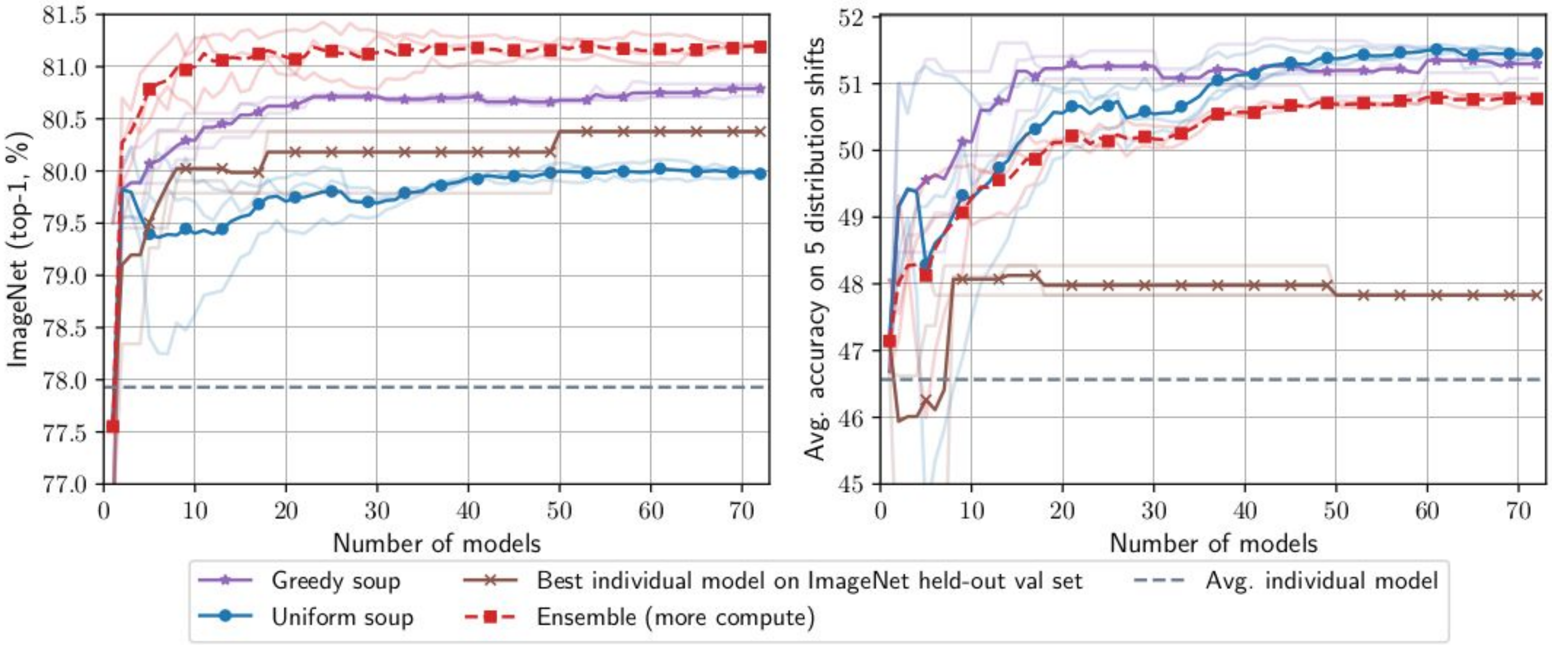


Figure B.1: For essentially any number of models, the greedy soup outperforms the best single model on both ImageNet and the out-of-distribution test sets. On the x-axis we show the number of models considered in a random search over hyperparameters while the y-axis displays the accuracy of various methods 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).

Soups performance on text classification task

Table J.1: Performance of model soups on four text classification datasets from the GLUE benchmark (Wang et al., 2018).

Model	Method	MRPC	RTE	CoLA	SST-2
BERT-base (Devlin et al., 2019b)	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 (Devlin et al., 2019b)	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 (Raffel et al., 2020b)	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 (Raffel et al., 2020b)	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 (Raffel et al., 2020b)	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**

Why averaging weights works

- The loss landscape of neural network training is non-convex with many solutions in different loss basins)=
- BUT: Recent work (Neyshabur et al., 2020) observes that fine-tuned models optimized independently from the same pre-trained initialization lie in the same basin of the error landscape

Loss visualisation

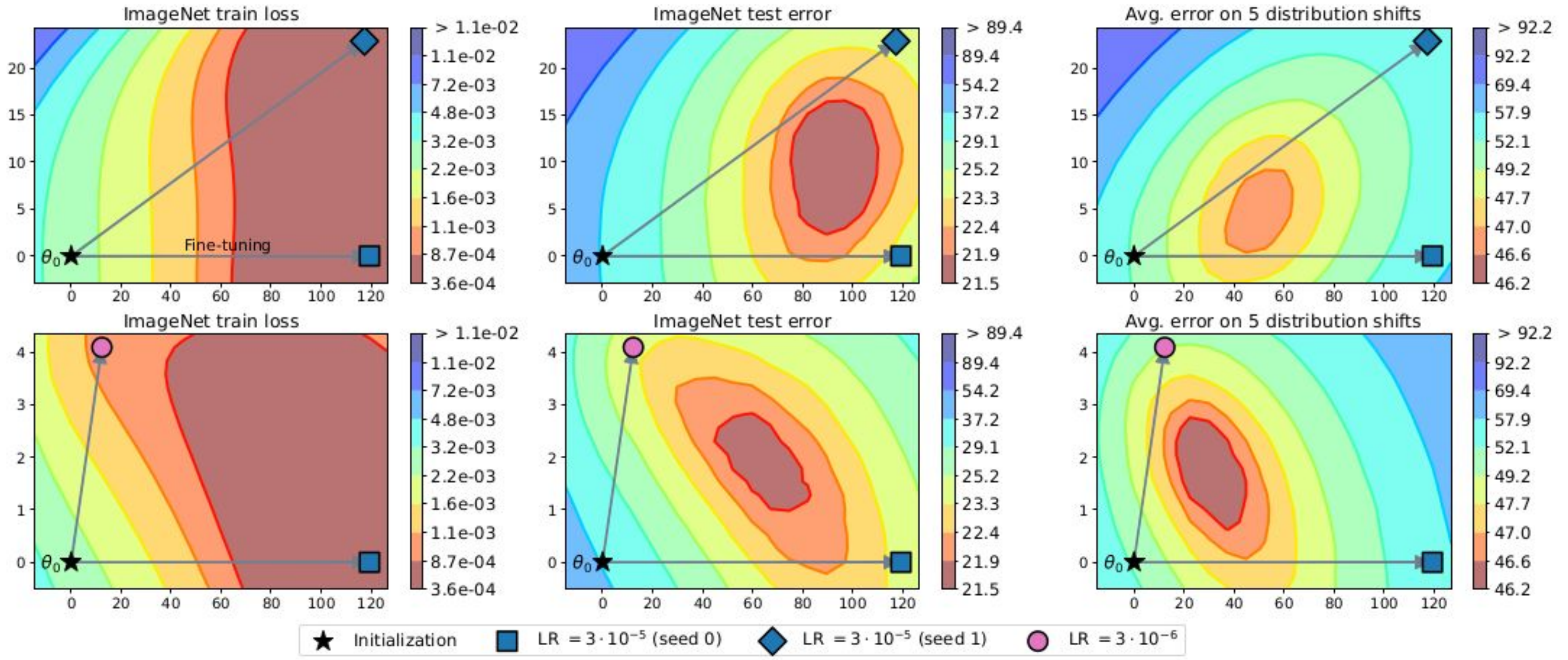
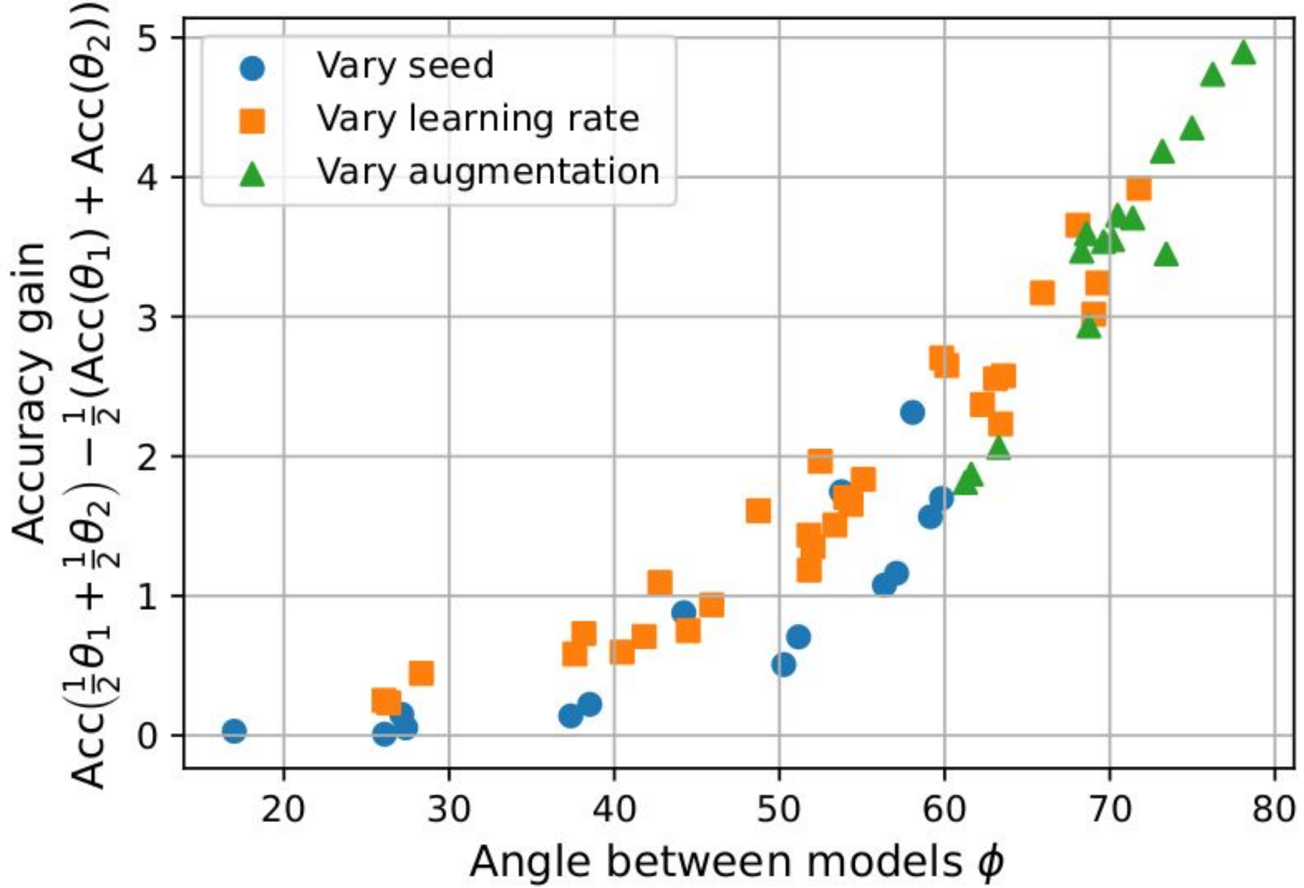


Figure 2: The solution with the highest accuracy is often not a fine-tuned model but rather lies between fine-tuned models. This figure shows loss and error on a two dimensional slice of the loss and error landscapes. We use the zero-shot initialization θ_0 and fine-tune twice (illustrated by the gray arrows), independently, to obtain solutions θ_1 and θ_2. As in Garipov et al. (2018), we obtain an orthonormal basis u_1, u_2 for the plane spanned by these models, and the x and y-axis show movement in parameter space in these directions, respectively.

Accuracy gain depending on angle between models



Loss visualisation

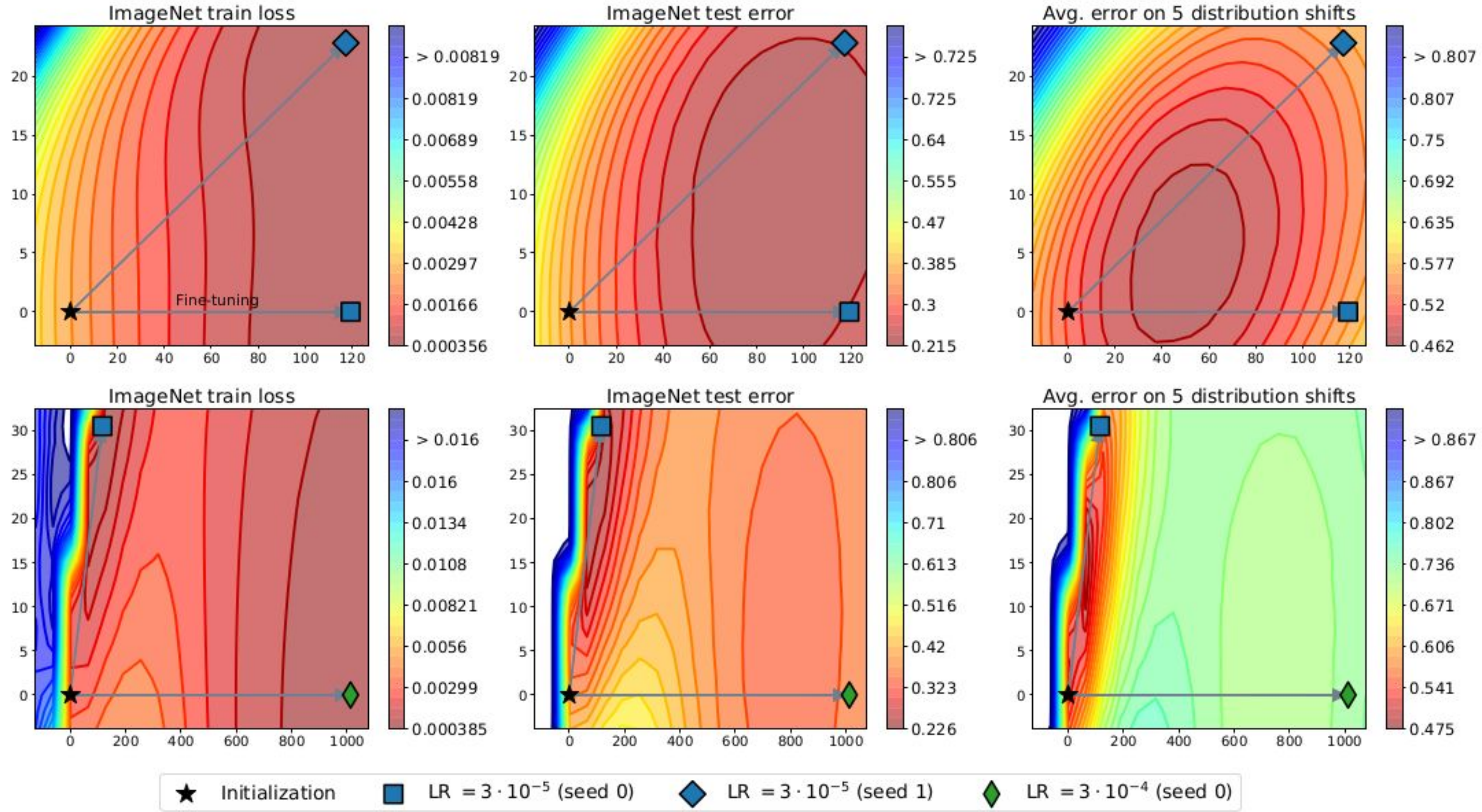
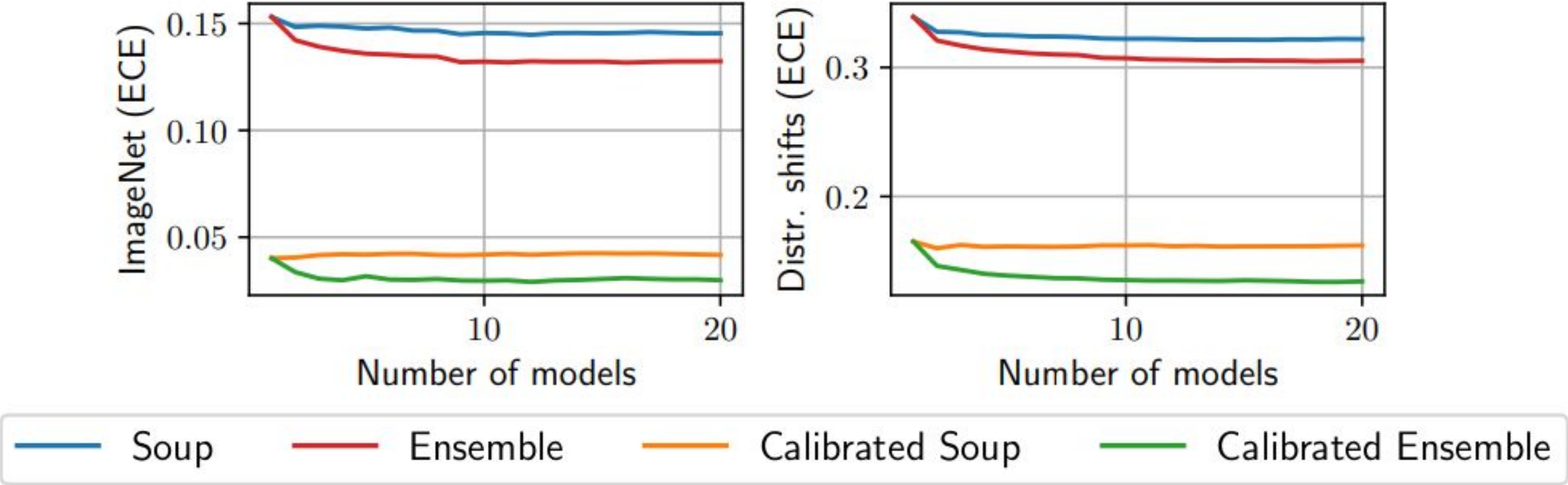


Figure J.1: Replicating Figure 2 with a 10x larger learning rate instead of 10x smaller in the second row.

Soups vs Ensembles

- May be less accurate than ensembles on in-distribution samples
- Don’t improve model calibration (ECE - expected calibration error)



Soups vs Ensembles

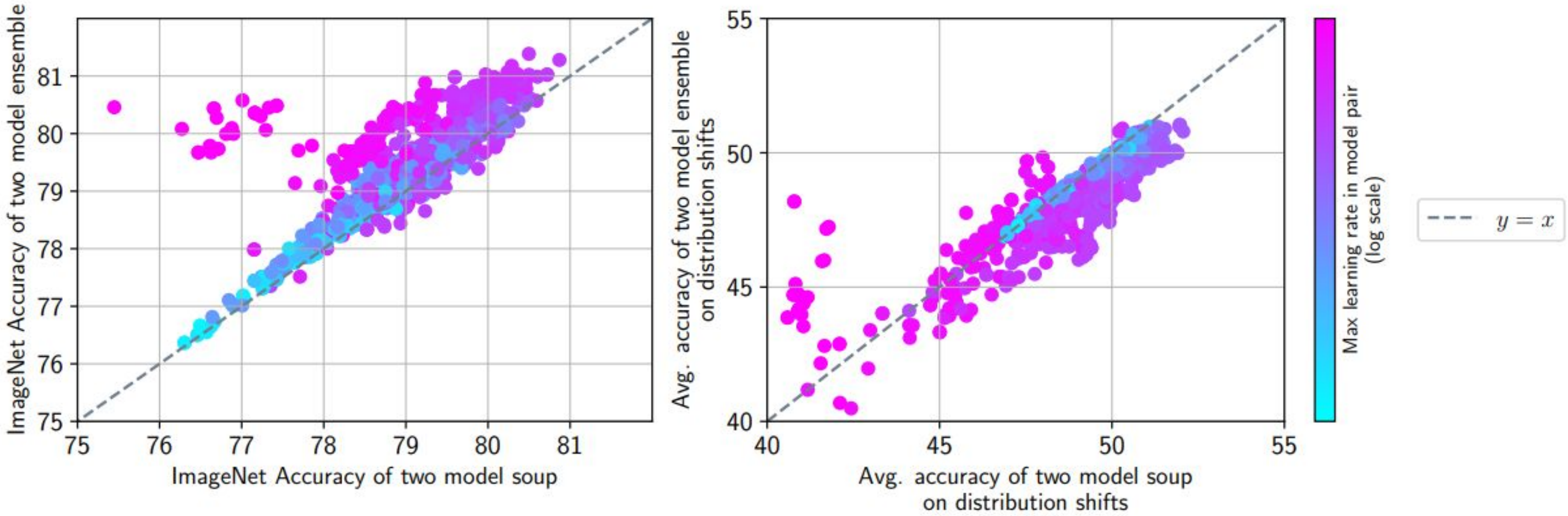


Figure 4: Ensemble performance is correlated with model soup performance. Each point on the scatter plot is a model pair with different hyperparameters. The x-axis is the accuracy when the weights of the two models are averaged (i.e., the two model soup) while the y-axis is the accuracy of the two model ensemble. Ensembles often perform slightly better than soups on ImageNet (left) while the reverse is true on the distribution shifts (right). Each model pair consists of two random greed diamonds from Figure 1.

Conclusion

- Soup is a way of building one final model using many fine-tined models by averaging weights
- Soups have O(1) inference
- Soups outperform single model in accuracy in image/text classification tasks
- Soups can perform not worse than ensembles (or even better in out-of-distribution samples)

References

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

- What is being transferred in transfer learning?