SELF-*Soup*ERVISION: COOKING AND SEASONING MODEL SOUPS WITHOUT LABELS FOR ADAPTATION

Anthony Fuller[1,3], **James R. Green**[1], **Evan Shelhamer**[1,2,3]
Carleton University[1] University of British Columbia[2] Vector Institute[3]

ABSTRACT

Model soups are strange and strangely effective combinations of parameters. They take a model (the stock), fine-tune it into multiple models (the ingredients), and then mix their parameters back into one model (the soup) to improve predictions. While all known soups require supervised learning, and optimize the same loss on labeled data, our recipes for Self-*Soup*ervision generalize soups to self-supervised learning (SSL). Our Self-Souping lets us flavor ingredients on new data sources, e.g. from unlabeled data from a task for transfer or from a shift for robustness. We show that Self-Souping on corrupted test data, then fine-tuning back on uncorrupted train data, boosts robustness by +3.5% (ImageNet-C) and +7% (LAION-C). Self-*Soup*ervision also unlocks countless SSL algorithms to cook the diverse ingredients needed for more robust soups. We show for the first time that ingredients can differ in their SSL training. We cook soups of these ingredients (e.g. MAE, MoCoV3, and MMCR) by our Self-Seasoning, which *mixes without labels*.

1 INTRODUCTION: MORE SOUPS, LESS SUPERVISION

Model soups make several models (the ingredients) by independent fine-tunings initialized from a single model (the stock), then merge them back into one model (the soup) by mixing parameters to improve prediction accuracy (Wortsman et al., 2022). Each fine-tuning varies in its configuration (e.g. optimization hyperparameters) and each mixing can be a simple average or more sophisticated linear combination. In this way, soups convert more training time into more accuracy without more inference time: the soup model needs only as much computation as the original model.

Model soups are surprisingly possible, in that mixing model parameters is absolutely not guaranteed to result in a better model (or even an equally good model!). They are also surprisingly productive with improvements across many settings: vision (Jain et al., 2023; Wortsman et al., 2022), language (Ablin et al., 2025; Jang et al., 2023; Chronopoulou et al., 2023), text-to-image (Biggs et al., 2024), federated learning (Chen et al., 2024), domain generalization (Ramé et al., 2023; Ramé et al., 2022), and class imbalance (Aminbeidokhti et al., 2025). However all known soups have to train ingredients by *supervised* learning—depriving our palettes of tasty new soups for many occasions.

We thus introduce Self-Soups, which are model soups made from ingredients that differ in their independent *self-supervised* training runs. Self-Souping vastly expands the menu of possible soups by harnessing different losses to flavor ingredients from different distributions without requiring labels (Fig. 1). We can "inter-train" models to make ingredients by self-supervised learning (SSL), after pre-training but before fine-tuning to a task, to enable transfer and robustness by optimizing more losses on more data. For example, we can inter-train on the *test* distribution to make shift-aware ingredients that improve robustness. In our first experiment, we inter-train on corrupted data (ImageNet-C (Hendrycks & Dietterich, 2019) and LAION-C (Li et al., 2025)), then fine-tune back on the distribution for which labels are available (ImageNet training data (Russakovsky et al., 2015)) to boost accuracy by +3.5 and +7%. In our second experiment, we mix ingredients that differ only in their SSL runs—made possible by Self-*Soup*ervision. For each VTAB dataset, we run self-supervised inter-trainings that differ in their self-supervised algorithms and algorithmic hyperparameters. We then mix these purely self-supervised ingredients by quickly "seasoning" (Croce et al., 2023) them: choosing the mixture conditioned on few-shot labels. We even mix soups for a task without training labels by a new and fully unsupervised variant that we call Self-Seasoning.

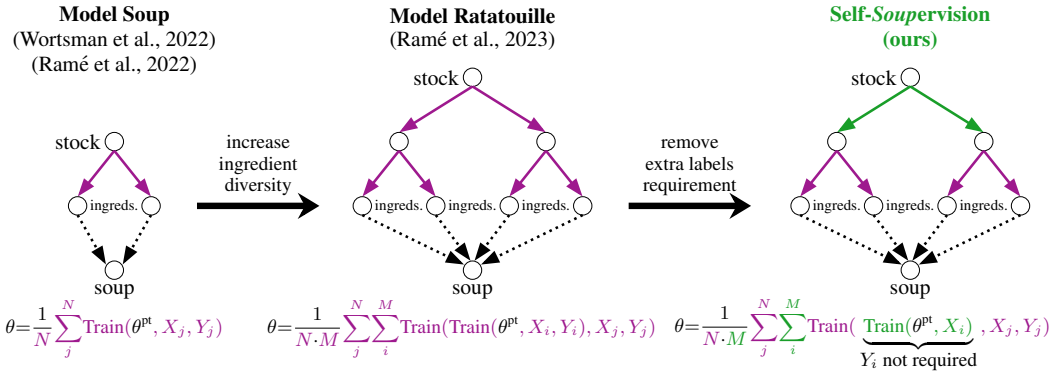

Figure 1: **Soups & Supervision.** Soups fine-tune then mix models to improve predictions. The original Model Soup (left) fine-tunes across hyperparameters for the task, while Model Ratatouilles (center) first inter-trains on *different labeled data* then fine-tunes for the task. Both are supervised and cannot harness unlabeled data. Our Self-*Soup*ervision (right) inter-trains across different losses and data to make more and better soups with and without labels. Our Self-Soups fine-tune then mix into a supervised model for the task. θ^{pt} is the pre-trained stock for initialization, N is the number of fine-tunings on data j, M is the number of inter-trainings on data i, and θ is the final model. We color **supervised** and **self-supervised** training runs / components. "ingreds." is short for ingredients.

2 Background: Supervised Soups

Initializing cooking with a stock. Soups require that the models for mixing (the ingredients) share the same initial parameters for optimization (the stock).

Adding ingredients by fine-tuning. Each fine-tuning of the stock creates an ingredient for mixing a soup. Multiple *different* fine-tunings are key for different ingredients: soups rely on differences among the ingredient models for their gains. To vary their ingredients, Model Soups (Wortsman et al., 2022) vary the optimization parameters, such as the learning rate and data augmentation.

Boosting ingredient diversity via inter-training. Model Ratatouille (Ramé et al., 2023) fine-tunes in two stages, optimizing ingredients for longer, and increasing their diversity for domain generalization. Ratatouille first initializes with a stock, then "inter-trains" on up to 5 auxiliary labeled datasets independently, and finally fine-tunes these models on the target task for mixing.

3 Method: Cooking without Labels and Instant Seasoning

We introduce **Self-*Soup*ervision**, which creates ingredients (parameters to mix) by training from a stock (parameters to initialize) without requiring labels at every stage. Our framework is broad; any soup that is made using ingredients that differ in their SSL runs qualifies as a Self-Soup; thus, there are endless possible instantiations of Self-*Soup*ervision. For example, the choice of stock, SSL algorithm and algorithmic hyperparameters, training data, training length/schedule, optimization hyperparameters, additional training stages (e.g. fine-tuning), etc. Centrally, Self-*Soup*ervision allows for cooking soups on more data and from new sources—e.g. that are closer to the target distribution—and using different losses—e.g. that are more aligned with the target task. Formally, we define Self-*Soup*ervision as:

$$\theta = \frac{1}{N \cdot M}\sum_{j}^{N}\sum_{i}^{M} \text{Train}(\overbrace{\text{Train}(\theta^{\text{pt}}, X_i)}^{Y_i \text{ not required}}, X_j, Y_j) \tag{1}$$

where θ are soup parameters, θ^{pt} are pre-trained / stock parameters, N is the number of fine-tunings per inter-training, M is the number of inter-trainings, X_i/X_j are the inputs of the i^{th}/j^{th} dataset, and Y_j are the labels of the j^{th} dataset. **Supervised** ingredients alone make standard soups. We

Table 1: **Self-*Soup*ervision on the test distribution provides large gains: +3.5% on IN-C and +7% on LAION-C.** These soups mix in ingredients inter-trained by SSL on *unlabeled* test data. In the disjoint setting, we use even-indexed test samples for inter-training on corruptions and odd-indexed samples for testing on corruptions. In the joint setting, we use even-indexed test samples for both inter-training and testing. "IN" is for ImageNet, "LA" is for LAION. Results % top-1 acc.

Reference (IN-Train)

Method	IN-Val	IN-C	LA-C
Supervised Soup	79.05	30.91	22.05
Cont. SSL + Soup	79.23	31.24	21.61
Self-Soup	79.11	32.23	22.40

Test-set inter-training

Method	Disjoint			Joint		
	IN-Val	IN-C	LA-C	IN-Val	IN-C	LA-C
Self-Soup (IN-C)	**78.96**	**35.72**	23.28	**78.96**	**36.02**	23.51
→ Best ingredient	78.91	31.41	19.55	78.91	31.70	19.57
Self-Soup (LA-C)	78.87	32.58	**29.48**	78.87	32.51	**30.32**
→ Best ingredient	78.87	28.25	23.38	78.87	28.09	23.84
Self-Soup (IN-C + LA-C)	78.81	34.43	26.39	78.81	34.55	26.83

introduce the **self-supervised** ingredients—which do not need labels—and which we can also use alone (i.e., by dropping the **supervised** ingredients) to make soups without fine-tuning to a task.

Mixing by Instant Seasoning. Seasoning (Croce et al., 2023) searches for a mixture over a grid of options by mixing each model, making predictions on a few-shot labeled dataset, and picking the best. While effective, this only applies to fine-tuned ingredients: seasoning makes predictions by mixing the classifiers. We instead mix purely self-supervised ingredients into a model for representation—rather than classification—then compute its representation on training and testing data for prediction by nearest neighbors. Our variant of seasoning mixes without classifier training for the "instant" seasoning of soups across different tasks. Specifically, we randomly sample the M mixture coefficients uniformly from the probability simplex.

Mixing by Self-Seasoning. We pair our new ingredients with a new and unsupervised way to mix: Self-Seasoning. We optimize our M mixture coefficients by gradient descent to minimize the entropy of predictions by nearest neighbors.

4 Experiments

Setup: Inter-train on shifts then fine-tune on ImageNet. We now allow for SSL inter-training on shifted data. In this case, SSL enables ingredients to learn from the test distribution *without labels* for robustness to it. We measure this on split data for optimization and evaluation (Tab. 1). We first inter-train 4 models (varying learning rates {1e-5, 2e-5, 3e-5, 4e-5}) by MAE (He et al., 2022) for 100K steps on unlabeled test samples that are *even-indexed*. We fine-tune these models back on the ImageNet training set for 10 epochs—doing this 4 times (varying fine-tuning learning rates {6e-5, 8e-5, 1e-4, 1.5e-4}). We report for *odd-indexed* and *even-indexed* test samples to measure accuracy when inter-training and eval samples are disjoint and joint, respectively.

Table 2: **Our Self-Soup cooked on ImageNet-C keeps its robustness advantage after test-time adaptation (TTA).** We run soups prepared on ImageNet-Train (from Tab. 1) on ImageNet-C (odd indices) with and without TTA—and repeat for our Self-Soup on ImageNet-C (*disjoint*, from Tab. 1). Results are % top-1 acc.

Method	without TTA	with TTA
Supervised Soup on ImageNet-Train	30.95	32.66
Self-Soup on ImageNet-Train	32.25	34.00
Self-Soup on ImageNet-C	35.72	37.50

Results: Shift-aware ingredients deliver robustness. Self-Souping on the test distribution (but not the test samples themselves) provides large gains: +3.5% on ImageNet-C and +7% on LAION-C. Self-Souping on test samples themselves provides a small boost on top: +0.3% on ImageNet-C and +0.8% on LAION-C. Despite the different types of corruptions present in ImageNet-C versus LAION-C, there are benefits to inter-training on one set of corruptions to the other set. Self-Souping over both test sets—i.e. where ingredients differ in their SSL inter-training data distributions and fine-tuning runs—keeps most of the shift-specific gains.

Bonus: Why not adapt to the shift at test time? Another unsupervised way to adapt to a shift is test-time adaptation (TTA), e.g. by minimizing prediction entropy (Wang* et al., 2021). We use SAR (Niu et al., 2023) as a SOTA TTA method. Our Self-Soup on ImageNet-C without SAR still

beats soups prepared on ImageNet-Train with SAR applied on ImageNet-C (Tab. 2). In this case, our soup made from shift-aware ingredients that we updated on the test distribution (odd-indexed), not test samples (even-indexed), outperforms models updated on test samples. This Self-Soup on ImageNet-C gains more with adaptation by SAR (35.72 → 37.50), showing the two strategies add.

Table 3: **Self-Seasoning (no training labels) is competitive with supervised seasoning (uses training labels).** Our Self-Seasoning finds mixture coefficients for our Self-Soups entirely without labels. For the best ingredient and seasoning runs, we select based on mini-VTAB training data—with labels. A uniform mix of our SSL ingredients provides a nice boost over the stock.

	Caltech101	CIFAR-10	CIFAR-100	DTD	Flowers102	Pets	Sun397	SVHN	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	dSpr-Loc-X	dSpr-Loc-Y	dSpr-Loc-Ori	KITTI-Dist	sNORB-Azim	sNORB-Elev
Ingredient	Self-Seasoning Coefficients																				
Stock	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MAE: default config	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MMCR: global-only	0.02	0	0	0.02	0.02	0.77	0.91	0	0	0.01	0.04	0	0	0	0	0	0	0	0	0	0
MMCR: global+local	0.78	0.98	0.99	0.8	0.81	0.07	0.09	0.87	0.98	0.99	0.95	0	0.99	0.98	0.77	0	0	0	0.74	0.77	0.76
MoCoV3: temp=0.1	0.14	0	0	0.17	0.16	0.03	0	0.07	0.02	0	0	0	0.01	0.01	0.19	0.15	0.99	0.97	0.21	0.13	0.15
MoCoV3: temp=1.0	0.06	0.01	0.01	0.01	0.02	0.13	0	0.05	0	0	0	0.99	0.01	0.01	0.03	0.85	0	0.03	0.04	0.1	0.09
Method	Method Top-1 % Accuracies																				
Stock	28.6	30.6	7.0	18.5	14.2	7.5	4.8	18.7	74.5	68.5	31.5	73.2	26.4	27.1	22.3	7.1	12.5	20.3	48.5	**11.3**	**24.7**
Uniform Soup	69.7	66.8	20.3	49.3	48.3	57.5	19.5	42.2	80.7	88.5	57.6	74.3	27.4	27.7	30.3	3.8	6.9	33.1	51.8	10.5	23.1
Best Ingredient	87.6	78.6	34.4	60.4	76.6	**71.0**	26.3	78.9	80.8	95.2	**74.2**	73.2	33.7	32.4	34.4	6.0	14.3	36.5	**53.6**	11.1	23.8
Seasoning	87.6	**80.9**	**37.6**	58.7	76.0	70.8	**27.3**	**79.0**	**81.5**	94.9	73.7	**74.8**	**34.6**	**33.8**	**35.7**	**11.6**	**17.9**	**37.4**	50.1	11.1	23.5
Self-Seasoning (ours)	**87.8**	76.9	34.1	**61.0**	**78.1**	67.1	23.6	77.6	81.1	**95.3**	73.3	73.9	17.5	29.9	26.4	4.1	7.3	33.2	49.2	9.5	19.0

Setup: Self-Seasoning of SSL ingredients. We now mix soups without supervised fine-tuning and evaluate by nearest neighbors (kNN). We make 6 ingredients per task with MAE, MMCR ($\times 2$) (Yerxa et al., 2023), MoCoV3 ($\times 2$) (Chen et al., 2021), and the stock. For the 5 task-specific ingredients, we train for 10K steps with a 4e-5 learning rate on the full training data for each VTAB task. To find mixture coefficients we Self-Season our Self-Soup by minimizing kNN entropy on the VTAB training data *without* labels (Zhai et al., 2020). We use k=16 for all tasks as it generally performs well. To put our Self-Seasoning in context, we do supervised seasoning to find mixture coefficients on the mini-VTAB's training data *with* labels. We also compare to the stock, a uniform mix of our ingredients, and the best ingredient based on kNN accuracy on the same *labeled* data.

Results: Self-Seasoning is competitive. Even without training labels for mixing, our Self-Seasoning is the most accurate on 4/21 tasks. Overall, seasoning with training labels is most effective, and outperforms the best ingredient—on dSpr-Loc-X it is almost twice as accurate.

5 Discussion

Related Work. Model stock (Jang et al., 2024) refines the mixing of soups with layer-wise reweighting using the angles between ingredient parameters—our method is compatible. Our inter-trainings each optimize their own model with a single self-supervised loss. Multi-task SSL instead jointly optimizes a shared model with multiple self-supervised losses (Doersch & Zisserman, 2017; Bachmann et al., 2022). Our inter-training on shifts is related to unsupervised domain adaptation (UDA): joint optimization on labeled "source" data and unlabeled "target" data (Saenko et al., 2010).

Limitations. Self-Soups enlarge the soup kitchen (SSL methods, hyperparameters, and data) with our new recipes, but there are more to cook. Our largest gains (+7%) need unlabeled target/shifted data, which may not be available. Other gains are modest ($\lesssim$1%) yet useful, as they do not raise inference costs. There are dozens of SSL algorithms absent from our study, but we choose from 3 different SSL families, so our findings may generalize within families.

Conclusion. We introduce Self-*Soup*ervision, which generalizes model soups to SSL. Self-Souping adds to the menu by harnessing different losses to flavor ingredients from different distributions without requiring labels. Self-Souping is most helpful when facing distribution shifts—and especially, when unlabeled shifted data is available for preparing ingredients. We also introduce Self-Seasoning, which learns ingredient mixtures for a task without training labels. We hope our recipes earn a spot in your cookbook and inspire new ones.

REFERENCES

Pierre Ablin, Angelos Katharopoulos, Skyler Seto, and David Grangier. Soup-of-experts: Pretraining specialist models via parameters averaging. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=MFNIka7nx0.

Masih Aminbeidokhti, Subhankar Roy, Eric Granger, Elisa Ricci, and Marco Pedersoli. LT-soups: Bridging head and tail classes via subsampled model soups. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tiEsJtw3FH.

Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. MultiMAE: Multi-modal multi-task masked autoencoders. In *European Conference on Computer Vision*, pp. 348–367. Springer, 2022.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Benjamin Biggs, Arjun Seshadri, Yang Zou, Achin Jain, Aditya Golatkar, Yusheng Xie, Alessandro Achille, Ashwin Swaminathan, and Stefano Soatto. Diffusion soup: Model merging for text-to-image diffusion models. In *European Conference on Computer Vision*, pp. 257–274. Springer, 2024.

Minghui Chen, Meirui Jiang, Xin Zhang, Qi Dou, Zehua Wang, and Xiaoxiao Li. Local superior soups: A catalyst for model merging in cross-silo federated learning. *Advances in Neural Information Processing Systems*, 37:20858–20886, 2024.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*, 2023.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Francesco Croce, Sylvestre-Alvise Rebuffi, Evan Shelhamer, and Sven Gowal. Seasoning model soups for robustness to adversarial and natural distribution shifts. 2023.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2051–2060, 2017.

Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*, 2019.

Samyak Jain, Sravanti Addepalli, Pawan Kumar Sahu, Priyam Dey, and R Venkatesh Babu. Dart: Diversify-aggregate-repeat training improves generalization of neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16048–16059, 2023.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *Proceedings of the European Conference on Computer Vision*, 2024.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*, 2023.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Kaggle and EyePacs. Kaggle diabetic retinopathy detection, 2015. URL https://www.kaggle.com/c/diabetic-retinopathy-detection/data.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

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.

Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pp. II–104. IEEE, 2004.

Fanfei Li, Thomas Klein, Wieland Brendel, Robert Geirhos, and Roland S. Zimmermann. Laion-c: An out-of-distribution benchmark for web-scale vision models. In *International Conference on Machine Learning (ICML)*, 2025.

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

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 7. Granada, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=g2YraF75Tj.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Alexandre Ramé, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In *NeurIPS*, 2022.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pp. 213–226. Springer, 2010.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European conference on computer vision*, pp. 516–533. Springer, 2022.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 210–218. Springer, 2018.

Dequan Wang*, Evan Shelhamer*, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Thomas Yerxa, Yilun Kuang, Eero Simoncelli, and SueYeon Chung. Learning efficient coding of natural images with maximum manifold capacity representations. *Advances in Neural Information Processing Systems*, 36:24103–24128, 2023.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark, 2020. URL https://arxiv.org/abs/1910.04867.

A Appendix

B VTAB References.

For clarity and credit, we reference the original datasets that went into the VTAB collection: Caltech101 (Fei-Fei et al., 2006), CIFAR-10/100 (Krizhevsky et al., 2009), DTD (Cimpoi et al., 2014), Flowers102 (Nilsback & Zisserman, 2008), Pets (Parkhi et al., 2012), Sun397 (Xiao et al., 2010), SVHN (Netzer et al., 2011), EuroSAT (Helber et al., 2019), Resisc45 (Cheng et al., 2017), Patch Camelyon (Veeling et al., 2018), Retinopathy (Kaggle & EyePacs, 2015), Clevr (Johnson et al., 2017), dSprites (Matthey et al., 2017), SmallNORB (LeCun et al., 2004), DMLab (Beattie et al., 2016), and KITTI (Geiger et al., 2013).

C More training details.

All runs. We always use: 0.01 weight decay, the AdamW optimizer (Loshchilov & Hutter, 2019), warmup for 10% of the steps and cooldown via cosine decay, and 3-Augment (Touvron et al., 2022) for data augmentation.

All SSL inter-training runs. For MAE inter-training, we initialize the decoder with the pre-trained MAE decoder. For MoCoV3 and MMCR inter-training, we use a simple 2-layer MLP as the projection head, we do not use an exponential moving average to compute target embeddings, and we warmup the head for only 1% of the steps (chosen so the head learns more quickly than the backbone). We choose these settings to keep it simple and do not tune them. Before mixing or fine-tuning the ingredients, we discard all algorithm-specific heads and only use the backbones/encoders.

All ImageNet fine-tuning runs. We fine-tune for 10 epochs on ImageNet-1K (following the original Model Soups (Wortsman et al., 2022)). We sweep learning rates {6e-5, 8e-5, 1e-4, 1.5e-4} with a 128 batch size. We always use LPFT, which initializes fine-tuning from the linear probed solution (Kumar et al., 2022).

Test-set inter-training. For model inter-training, we train for 100K steps with a 128 batch size using the default MAE settings (i.e. 75% masking ratio and 8 decoder layers), and sweep learning rates {1e-5, 2e-5, 3e-5, 4e-5}.

Test-time adaptation. We sweep base learning rates {1e-5, 3e-5, 5e-5, 8e-5, 1e-4, 3e-4, 1e-3, 3e-3}. A 5e-5 base learning rate is best. We use a 128 batch size, which sets the actual learning rate: $\text{lr} = (\text{base\_lr}/64) \cdot \text{batch\_size}$

D SEASONING.

For supervised seasoning, we try 1K random samples of mixture coefficients (Dirichlet distribution with concentration = 1), and pick the best on kNN accuracy on the mini-VTAB training set for each task. For the "best ingredient", we also pick it based on kNN training accuracy for each task. For Self-Seasoning, we train the mixture coefficients with AdamW for 100 epochs starting with a 0.1 learning rate and cosine-decay it to 0.01 with a 256 batch size (we did not tune this because it worked well enough). We initialize all 6 parameters to 0 and apply a softmax so the coefficients sum to 1. We use k=16 and a 0.07 temperature.

```python
def knn_inbatch_neighbor_entropy(Z, k=16, T=0.07):
    B = Z.shape[0]

    # L2 normalize embeddings
    Z = F.normalize(Z, dim=1, p=2)           # [B, D]

    # Compute pairwise cosine similarities
    S = Z @ Z.T                               # [B, B]
    S.fill_diagonal_(float("-inf"))           # mask self

    # Select top-k neighbors
    S_topk, _ = S.topk(k, dim=1)          # [B, k]

    # Compute neighbor distribution
    p = (S_topk / T).softmax(dim=1)           # [B, k]

    # Compute entropy
    H = -(p * p.clamp(min=1e-12).log()).sum(dim=1)

    return H.mean()
```