Adversarial Learned Soups

Brian Huang
branhung@mit.edu
MIT

Abstract

To make computer vision models more adversarially robust, recent literature has made various additions to the adversarial training process, from alternative adversarial losses to data augmentations to the usage of large numbers of diffusion-generated synthetic samples. However, models trained for adversarial robustness often face an inherent tradeoff between performance on clean images and performance against adversarial attacks. Methods that primarily seek to boost adversarial robustness may not optimize for the best combined performance along the clean-vs.-adversarial tradeoff. We devise a method to finetune adversarially trained models for combined clean and robust performance, borrowing from the method of "model soups," where parameters within an ensemble of finetuned checkpoints are averaged to form new model weights. Such model soups have been shown to improve performance in transfer learning settings while maintaining or improving the original task performance; extending from this observation, we find that linear interpolation of adversarially robust ensemble parameters reaps similar benefits in the tradeoff between robustness and clean accuracy. Furthermore, we construct a wrapper architecture, or "learned soup," to adversarially train our interpolation coefficients for model soups, and find that, in some cases, directly training the souping coefficients leads to a more robust model than grid-searching for the coefficients. This method of *adversarial learned soups* can be applied in conjunction with existing methods for adversarial training, further bolstering the current arsenal of defenses against adversarial attacks.

1 Introduction

Deep neural networks are notoriously brittle. In the realm of computer vision models, one prime example of brittleness is the phenomenon of adversarial attacks, where clean image inputs can be perturbed within small, human-imperceptible distortion bounds to severely deteriorate model performance, often to near-zero accuracy. In the years since adversarial attacks were first elucidated in [IJC15] [CWI+14] [ND17], various approaches to making models robust to adversarial attacks have been developed in the literature. Among the most effective defenses is adversarial training (AT) [AAL+18], in which models are trained on a gradient-based attack instead of clean inputs. While models treated with some degree of adversarial training–whether fully trained with AT, finetuned with AT from a clean pretrained checkpoint, or trained with a fixed proportion of clean vs. adversarially perturbed images [HYJ+19]–boast impressive robustness against a diverse range of attacks [FM20b], they are often accompanied by a degradation in performance on the original task, i.e. accuracy on clean images. Indeed, vulnerability to adversarial attacks is an inherent, fundamental flaw of deep neural networks: there is a provable tradeoff between robustness and clean performance that cannot be overcome even in the limit of infinite data [DSL+19].

Adversarial training has stood the test of time against recent and more powerful attacks [FM20b] [FM20a], so a large proportion of recent research in adversarial robustness for computer vision models has centered around modifications and augmentations to the adversarial training process. Works such as MART [YDJ+20] and Local Linearity Regularization [CJS+19] focus on modifying the objective function used in adversarial training. Other works such as [SSO+21] find that additional synthetic data, to the tune of millions of diffusion-model-generated images, lead to impressive gains in robustness, an insight that has led to the current state-of-the-art in adversarial robustness [FMV+21].

Often, across this wide range of methods, the emphasis lies mostly on improving adversarial robustness without also prioritizing clean performance. Because adversarially robust models tend to lose significant

clean accuracy in gaining robustness, the best-performing robust models still lag significantly behind in clean accuracy compared to state-of-the-art-performing models that have not undergone adversarial training. As such, there is still a pressing need for methods that manage this fundamental tradeoff between clean performance and adversarial robustness. For example, when deploying computer vision models in real-world settings, it may fit practical requirements better to maximize robustness given a threshold that clean accuracy should satisfy, rather than employ the model with the absolute highest robust accuracy.

Recently, [MGJ+22] [MGS+22] [GMS+22] have elucidated a weight-space interpolation method, dubbed "model soups," for improving the combined performances of models on a target task and on distribution-shifted tasks. In [MGJ+22], the authors averaged the parameters between a trained model and one checkpoint finetuned from that model; the authors expanded on this method in [MGS+22], instead averaging many models finetuned with different hyperparameters from the same pretrained model. For both setups, the averaged model often sees gains in target task accuracy and/or distributional robustness over any of the individual, "ingredient" models.

Our contributions. Inspired by these advances, we adapt several variants of model soups to the adversarial robustness setting. We find that, in some cases, manual grid search of the interpolated parameters finds souped models with improved clean accuracy, robust accuracy, or both over any of the ingredient models. Manual grid search is most tractable for model soup settings with only two or three ingredient models, and reveals a smooth transition in clean accuracy and robust accuracy as the strength of interpolation moves between ingredient models. Alongside ways to vary the "strength" of AT, such as varying distortion bounds (epsilon in an Lp-norm bounded attack) or varying numbers of epochs, manual grid search can further tune the resulting tradeoff between clean and robust accuracy, or in some cases, even improve combined clean and robust accuracy beyond individual models finetuned with AT.

For settings with many ingredient models where manual grid search is intractable, we explore alternative methods of tuning model soups. The learned soup method, first mentioned in an appendix of [MGS+22], directly trains the interpolation coefficients between ingredient parameters. Starting from its initial appearance in [MGS+22], we iterate on the training recipe for learned soups; we implement random initialization, a tuned learning rate schedule, and a bespoke dropout analog that we dub *detach dropout*. Building on this recipe, we adapt AT to the training process for learned soups. This adversarial learned soup smoothly trades off between clean accuracy and robust accuracy for models, and the level of tradeoff can be controlled by AT hyperparameters such as the epsilon in the Lp-norm attack.

We evaluate our adversarial learned soup and grid search on a wide range of architectures and AT hyperparameter variations. We show that the clean-robust tradeoff-balancing behavior of adversarial learned soups behaves consistently across many variants of ResNets and WideResNets; we also integrate the adversarial learned soup into existing recipes for adversarial training that use TRADES [HYJ+19] or a large synthetic dataset [SSO+21] [VST+22] [ZTC+23], demonstrating that our learned soup can be used in tandem with current state-of-the-art methods within adversarial training without sacrificing the efficacy of these methods. Finally, we discover some cases of model architecture or AT hyperparameters where learned soups (and other model soups) lead to underperformance or even degenerate performance, illuminating some potential limitations of these model soup-based methods.

2 Preliminaries

At its core, this paper

- explores the interaction between model averaging (most directly, the model soups method) and various flavors of adversarial training,
- and devotes significant attention to iterating on the learned soups method introduced in [MGS+22], designing training recipes for learned soups that include a novel dropout mechanism we name *detach dropout*.

In this section, we provide preliminary knowledge about adversarial training and model soups, and our implementation of these methods. In the following section, we explain architectural and experimental details about our *adversarial learned soup*.

Adversarial training. We create adversarially robust models via the adversarial training method first elucidated in [AAL+18]. Let $\mathcal{D}$ denote the data distribution from which we sample image-label pairs $(x, y) \sim \mathcal{D}$; any such (x, y) constitute *clean* data. To introduce adversarial data, let Δ denote the space of allowed perturbations under some threat model; for example, using the common threat models of L_∞-bounded or L_2-bounded perturbations, Δ respectively constitutes the L_∞-box or L_2-ball of radius ϵ, where epsilon is a hand-selected upper bound on the allowed perturbation distance. With respect to a loss function L and a model with parameters θ, an adversarial example from some input (x, y) can be generated as $(x + \delta, y)$, where δ satisfies $\arg\max_{\delta \in \Delta} L(\theta; x + \delta, y)$. Therefore, to train a model robust to adversarial examples, we aim to minimize the empirical risk for adversarial examples, written as a minimax problem

$$\min_{\theta} \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}} \left[\max_{\delta \in \Delta} L(\theta; x_i + \delta, y_i) \right] \tag{1}$$

where (x_i, y_i), $1 \leq i \leq n$ is our training set. In [AAL+18], this optimization problem is solved by replacing any data sample with an adversarial perturbation during the standard training loop, so that the model is always optimized against its own worst-case adversarial examples at every point in training. In practice, projected gradient descent is used to find perturbations in Δ that are approximately worst-case for usage in adversarial training or evaluation for adversarial robustness.

In addition to the AT procedure described above, we also perform experiments using TRADES [HYJ+19]. In our implementation of this method, we use both the clean data sample and an adversarial perturbation of the sample in training, and optimize on a weighted average of the losses from each. Cast as an ERM objective, we minimize

$$\min_{\theta} \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}} \left[\alpha L(\theta; x_i, y_i) + (1 - \alpha) \left(\max_{\delta \in \Delta} L(\theta; x_i + \delta, y_i) \right) \right] \tag{2}$$

where α is a hyperparameter controlling the desired "balance" of standard vs. robust optimization.

Model soups. For the model soups procedure, the first important step is selecting the range of hyperparameters or training configurations that are expected to generate several different finetuned checkpoints residing in the same loss basin. There are many potential hyperparameters to choose for variation in finetuning runs, including number of epochs, batch size, learning rate, weight decay, or even SGD noise, i.e. the random order of training batches in an epoch. For the adversarial robustness setting, our choices of hyperparameters for our experiments were the threat model and ϵ, which jointly specify the space of allowed perturbations Δ, as well as number of epochs, which controls clean and robust overfitting.

Let θ denote our pretrained parameters, corresponding to, for example, a ResNet or WideResNet trained with AT for hundreds of epochs. Then, for $t \in \{L_\infty, L_2, L_1\}$ and $\epsilon \in \mathbb{R}^+$, let $\theta(t, \epsilon)$ denote the parameters generated by finetuning from θ using AT with Δ specified by threat model t and perturbation bound ϵ. In our experiments, we mainly explore the finetuning settings of $\theta(L_\infty, 2/255)$, $\theta(L_\infty, 4/255)$, $\theta(L_\infty, 8/255)$, $\theta(L_\infty, 16/255)$, $\theta(L_2, 0.5)$, $\theta(L_1, 12)$, as well as clean finetuning denoted by $\theta(L_\infty, 0)$ (no allowed perturbations at $\epsilon = 0$). In addition to finetuning with variation in adversarial perturbation hyperparameters, we also vary number of epochs by performing one longer adversarial finetuning run and saving intermediate checkpoints at regular intervals throughout the run.

In each of the above experiments, with the "ingredient" finetuned checkpoints generated, we explore several approaches to model averaging between our ingredients. Let $\theta_1, ..., \theta_n$ denote our checkpoints which will be averaged to produce some checkpoint θ_{soup}. The baseline model soup comes from the non-weighted, uniform average between all parameters, $\theta_{soup} = \frac{1}{n} \sum_{i=1}^{n} \theta_i$; this is referred to as the "uniform soup" in [MGS+22]. Another core method in [MGS+22] is a "greedy soup," which accumulatively includes the best-performing individual models in the soup until soup performance stops increasing; because the "stopping rule" of greedy soups relies on a validation set, and we cannot split off a validation set without data contamination with respect to our pretraining, we don't feature the greedy soup method at the forefront of our experiments.

In lieu of greedy soups, we perform a grid search for weighted averages of model parameters when we only use two or three finetuned checkpoints; here, we evaluate all possible combinations of soups $\theta_{soup} = \sum_{i=1}^{n} \alpha_i \theta_i$, where the interpolation coefficients α_i are chosen with constraints $\sum_{i=1}^{n} \alpha_i = 1$, $\alpha_i \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. This is equivalent to the averaging method used in [MGJ+22] and in [FSE+23].

A more general formulation of model soups comes from averaging model parameters "by-layer," or having independent interpolation coefficients for each layer of the ingredient models. For an architecture with L layers, let $\theta_i^{(\ell)}$ denote the parameters of layer ℓ in checkpoint i, and let $\alpha_i^{(\ell)} \in [0,1]$ be the corresponding averaging coefficient for $\theta_i^{(\ell)}$ for our model soup. Then we can devise a model soup

$$\theta_{soup} = \{ \sum_{i=1}^{n} \alpha_i^{(\ell)} \theta_i^{(\ell)} \mid 1 \leq \ell \leq L \}.$$

The search space of all possible $\alpha_i^{(\ell)}$ is much larger than what is tractable via manual grid search; thus, to find optimal coefficients for by-layer model soups, we employ the learned soup method.

3 Learned Soups

To train interpolation coefficients for a model soup, we construct a learned soup architecture which operates as a "wrapper" around a series of finetuned models. For each layer ℓ of the finetuned models, we instantiate a softmax layer, whose outputs function as the respective interpolation coefficients $\alpha_i^{(\ell)}$. Softmax is chosen in order to satisfy the constraints $\alpha_i^{(\ell)} \in [0,1] \ \forall \ i$ and $\sum_{i=1}^{n} \alpha_i^{(\ell)} = 1$. At any forward pass through the learned soup, the average between finetuned checkpoints, weighted using the softmax outputs $\alpha_i^{(\ell)}$, is computed and used for the input, and the resulting gradient is backpropagated only through the learned soup softmaxes. The first design and implementation of this learned soup can be seen in Appendix I of [MGS+22]. We reimplement our learned soup from scratch, using *einsum* and *torch.func.functional_call* to compute the forward pass efficiently. An illustration of the learned soup architecture is given in Figure 1.

The implementation in [MGS+22] uses a constant initialization for the softmax layers, i.e. the learned soup is initialized as a uniform soup. Empirically, when training our learned soup on non-adversarial tasks, we find that learned soups get "stuck" at the uniform initialization, so that even after 10 epochs of training, the interpolation coefficients deviate very little from the uniform soup. Because greedy soups, where interpolation coefficients for many of the ingredient checkpoints are zero, consistently outperformed uniform soups in [MGS+22], this phenomenon of getting stuck at uniform initialization must be a suboptimal behavior. Furthermore, when training for more epochs until interpolation coefficients diverge from uniform initialization, we see that this divergence happens only at later layers, and earlier layers tend to remain stuck at uniform initialization. This behavior is an artifact of the sequential softmaxes that constitute the learned soup architecture, which often lead to vanishing gradients at early layers.

To remedy these problems, we introduce several changes or new methods to learned soup training. We change the initialization from uniform to Xavier [XY10] to eliminate getting stuck at initialization. In many cases, as shown by greedy soup examples in [MGS+22], an optimal or near-optimal solution for interpolation coefficients occurs when some coefficients are zero, which requires very large negative weights pre-softmax for our learned soup layers; as such, we adopt SGD with a cosine learning rate scheduler, and employ very large base learning rates ($lr = 10$) to encourage large movements in pre-softmax learned soup weights after only a few epochs.

Lastly, to remedy the vanishing gradients problem, we randomly detach some of the averaged layers during a forward pass, with probability of detachment controlled by a hyperparameter. In our learned soup architecture, an averaged layer contains a batch matrix product between the softmax interpolation coefficients and the checkpoint layer parameters as part of its computational tree, so backpropagation must pass through the softmax for that layer, leading to vanishing gradients after many sequential softmaxes. When an averaged layer is detached before dropout, the softmax weights are no longer updated; in exchange, the backpropagation no longer passes through that layer's softmax, and the remaining, non-detached layers face much fewer sequential softmaxes in the backpropagation up to them, eliminating the vanishing gradient problem. This trick, in which some learned soup softmax weights are randomly "pruned" during training, bears resemblance to the dropout method traditionally used in deep learning; hence, we refer to this method as *detach dropout*.

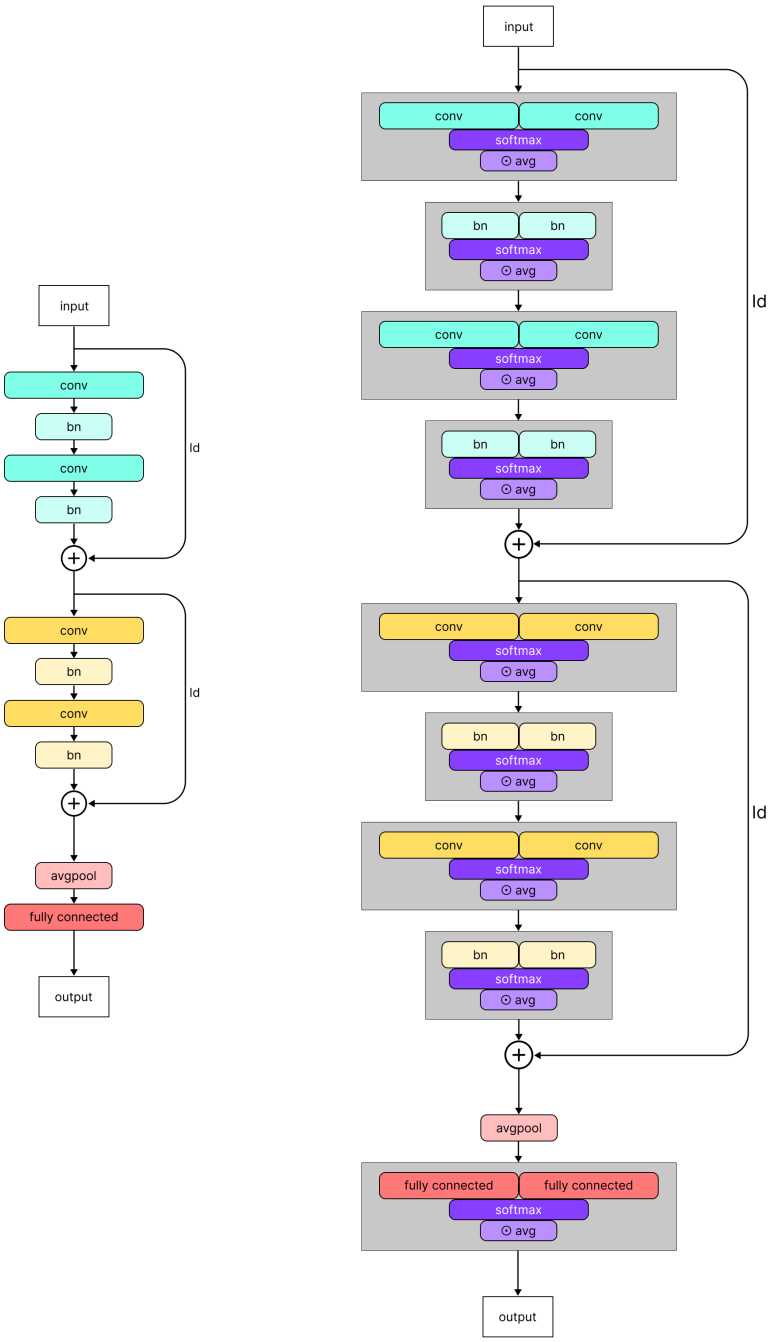


Figure 1: We give an example of a small ResNet architecture on the left, accompanied by our architecture of a learned soup between 2 checkpoints of the ResNet model. An averaged layer is computed from a list of the corresponding layer from each checkpoint and an accompanying 1-D softmax tensor, containing the interpolation coefficients; the $\odot$ is used to denote the "batch matrix product" that produces the weighted average. We call attention to which layers are averaged and which are not. For example, notice that batch-norm parameters are actually averaged in our implementation (and in the original model soups [MGS+22]); all named parameters in a PyTorch module are averaged, while buffers are not.

4 Experiments and Results

We perform model soup experiments on pretrained and finetuned checkpoints derived from a comprehensive range of models. For toy experiments, we pretrain a ResNet-9 and ResNet-50. Separately, we experiment with several recent models on RobustBench [FMV+21], including some with state-of-the-art performance as of August 2023. Our main experiments are performed on the following models:

- A WideResNet-28-10 trained with TRADES on a mix of real CIFAR-10 data and synthetic data generated from the diffusion model in [TMT+22]. This model was developed in [ZTC+23], and achieves the third-highest robust accuracy against L_∞, $\epsilon = 8/255$ attacks out of all models on RobustBench as of August 2023. For clarity we refer to this model as the WideResNet from Wang et al.
- A WideResNet-28-10 trained with AT on a proportion of Gaussian-sampled synthetic data, obtaining 58.63% accuracy on the RobustBench attack suite. This is the model used in the main results of [FSE+23]; we experiment on the same model to reproduce, and benchmark against, the results of [FSE+23]. For clarity we refer to this model as the WideResNet from Gowal et al.

For the ResNet-9 and ResNet-50, we pretrain with AT on CIFAR-10 for 150 epochs. During training we employ AutoPGD from [FM20b], a stronger attack than standard pgd; we use the threat model with L_∞ $\epsilon = 8/255$, and perform 2 iterations of pgd for each attack for faster training. We set the train and test batch size to 128 and optimize via SGD with momentum $= 0.9$, with no weight decay. We use a multi-step scheduler which sets the learning rate to our chosen base value 0.1 for the first 75 epochs, then shrinks the learning rate to 0.01 at epoch 75 and 0.001 at epoch 113. Crucially, we employ earlystopping based on validation loss, which significantly boosts the performance of standard AT as found in [LEJ20].

When finetuning the ResNet-9, ResNet-50, and WideResNet from Gowal et al., we use a cosine annealing scheduler with base learning rate 0.001. We employ various threat models of AutoPGD; to reiterate the Preliminaries section, we include L_∞ attacks with different epsilons $\epsilon = \{2/255, 4/255, 8/255, 16/255\}$, the L_2 $\epsilon = 0.5$ attack, and the L_1 $\epsilon = 12$ attack. Our batch size and optimizer are the same as in ResNet pretraining, but this time we turn off earlystopping. We finetune for 10 epochs to generate a grid of finetuned checkpoints with varying threat models and distortion bounds; these finetuning settings match those used in [FSE+23], except that we use a much smaller base learning rate. For our grid of intermediate checkpoints, we finetune with the L_∞ $\epsilon = 8/255$ attack for 50 epochs, and save intermediate checkpoints at epochs 10, 20, 30, and 40, resulting in 5 finetuned checkpoints with varying numbers of epochs along the same optimization pathway.

We mainly experiment on three configurations of learned soups and grid-searched soups:

- between varying L_p, with ingredients $\theta(L_\infty, 8/255)$, $\theta(L_2, 0.5)$, $\theta(L_1, 12)$. We refer to these as "threat model soups."
- between varying ϵ, with ingredients $\theta(L_\infty, 4/255)$, $\theta(L_\infty, 8/255)$, $\theta(L_\infty, 16/255)$. We refer to these as "eps soups."
- between intermediate checkpoints, with ingredients $\theta_{10}(L_\infty, 8/255)$, $\theta_{20}(L_\infty, 8/255)$, $\theta_{30}(L_\infty, 8/255)$, $\theta_{40}(L_\infty, 8/255)$, $\theta_{50}(L_\infty, 8/255)$, where the subscript denotes number of epochs in finetuning. We refer to these as "continuous soups."

When training learned soups for all models, we use a cosine learning rate with base lr 10 and a detach dropout probability of 0.8. We train for a number of epochs roughly proportional to the number of ingredients; for learned soups on varying L_p or ϵ, we train for 30 epochs, and for learned soups between intermediate checkpoints, we train for 50 epochs. We demonstrate clean training, AT, and TRADES with varying ratios for our learned soups.

Generally, when finetuning for soup ingredients and training learned soups, we incorporate all AT methods and augmentations used in pretraining if possible. This rule guides our training processes for our large models. For the WideResNet from Wang et al., the authors provide publicly available GitHub repositories for replicating their training and evaluation results; we load pretrained checkpoints from RobustBench and train using the authors' configuration files and synthetic datasets.

All experiments are performed on CIFAR-10.

4.1 Manual grid search

We perform a grid search for model soups by evaluating all combinations of interpolation coefficients α_i along a fractional "step size" between 0 and 1. Our step size for all experiments is 0.2, i.e. all alphas $\alpha_i \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. (With two ingredients, this constitutes 6 soups, and with three ingredients, this constitutes 21 soups.) The grid search results on our ResNets are shown in Figure 2. Our soups smoothly trade off performance between clean and adversarial tasks, but unlike in [MGJ+22], averaging models generally does not improve absolute performance above the finetuned checkpoints. Except for L_2 robust accuracy for ResNet-9 threat model soups and clean accuracy for ResNet-50 continuous soups, the best accuracy of any variant generally occurs on the endpoint of the grid search, i.e. for one of the finetuned checkpoints.

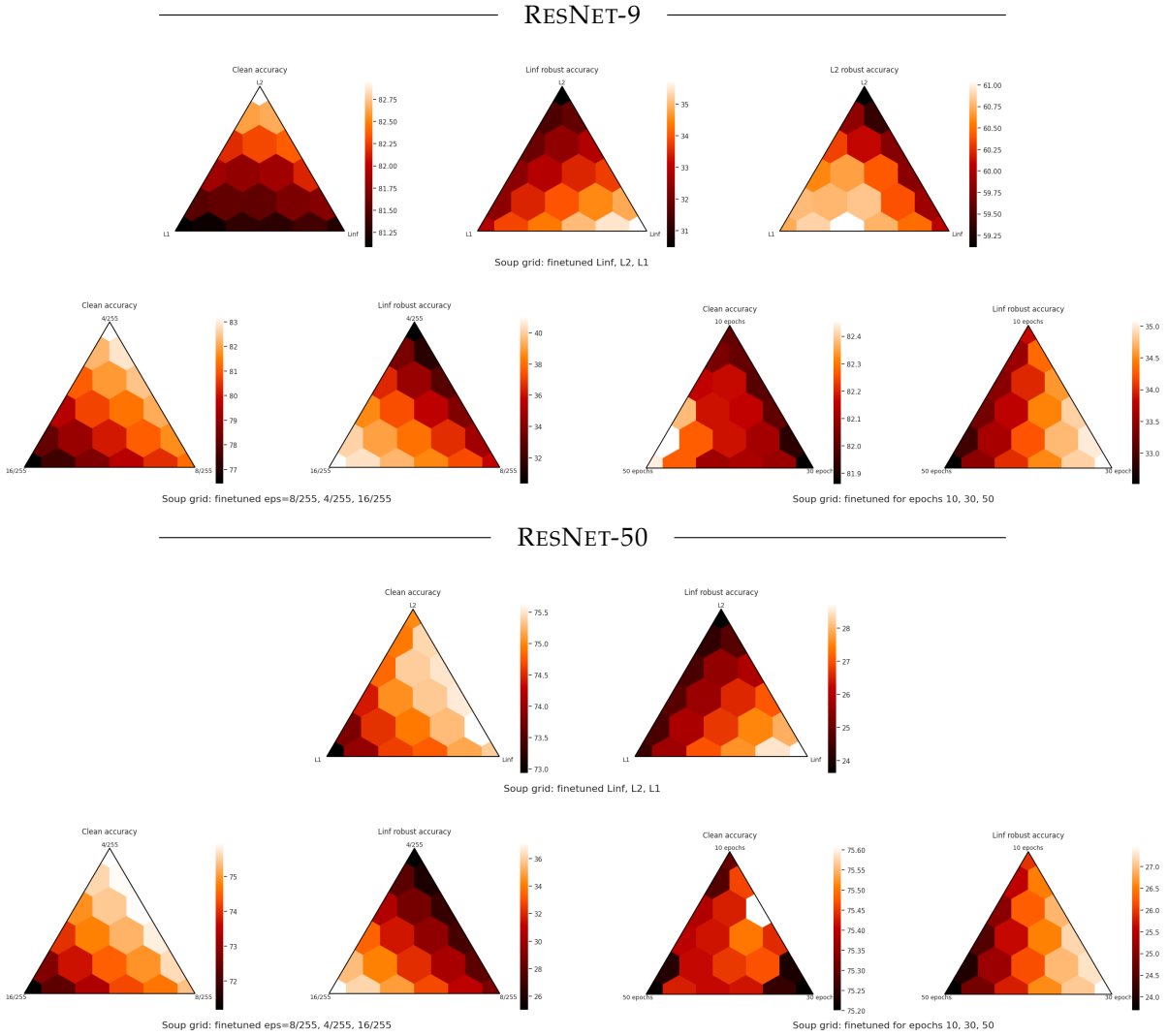


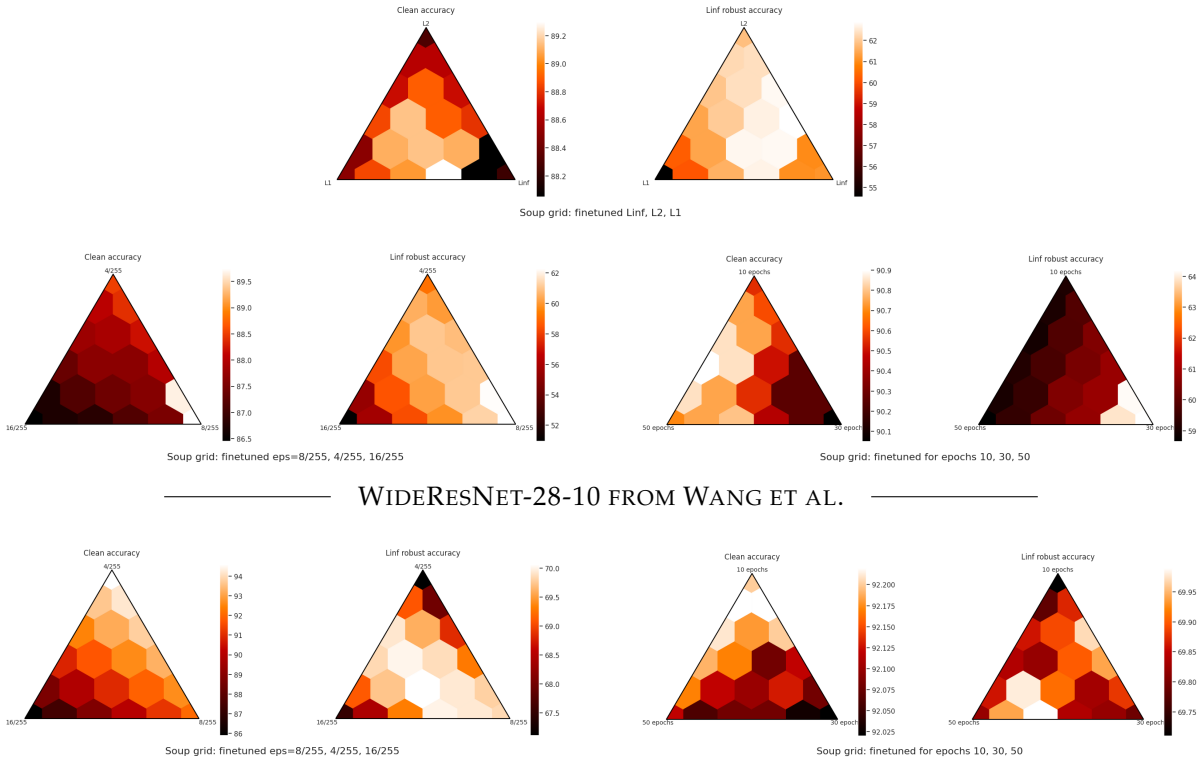
Figure 2: Ternary plots for grid search of weighted averages of finetuned checkpoints. We show accuracies on the full CIFAR-10 test set. The robust accuracy is evaluated using a strong L_∞ $\epsilon = 8/255$ AutoPGD attack with 100 iterations of pgd. The L_2 accuracy is similarly evaluated on a L_2 $\epsilon = 0.5$ AutoPGD attack.

Previous research on model averaging, across works such as [JGD+20] [BHC20], has found a convex loss basin in the shared region of finetuned models from a pretrained checkpoint; for our ResNet soups, the clean and robust accuracies as a function of interpolation coefficients appear mostly convex, but there are some hints of non-convexity, such as in the clean accuracies for ResNet-50 continuous soups. In addition, accuracies of ResNet soups generally trend with the strength of the attack in AT, as well as whether the threat model at training matches the threat model at evaluation. For example, L_∞ robust accuracy increases

with higher interpolation coefficient assigned to the L_∞-finetuned checkpoint in both threat model soups, and with higher interpolation coefficient assigned to the longest-trained, 50-epoch model in both continuous soups. In both eps soups, clean accuracy is best for the endpoint corresponding to the $\epsilon = 4/255$, the smallest distortion bound in the set.

Extending our grid search to the WideResNets from Gowal et al. and Wang et al., the resulting behavior of our soups no longer follows these intuitions. Results are shown in Figure 3. Clean and robust accuracies do not strongly trend with the strength or threat model of attack in finetuning. In addition, many regions of finetuned checkpoints have an "accuracy landscape" which is highly non-convex, visible even for our relatively coarse-grained grid search; this is especially apparent in the experiments on the Wang et al. WideResNet. However, as opposed to our ResNets, our WideResNets consistently see improvements in absolute clean and/or robust accuracy from model averaging. For every soup configuration except for the eps soup on Gowal et al., the highest clean or robust accuracy occurs near the midpoint of the grid search, where all finetuned checkpoints are significantly averaged into the soup.

WIDERESNET-28-10 FROM GOWAL ET AL.



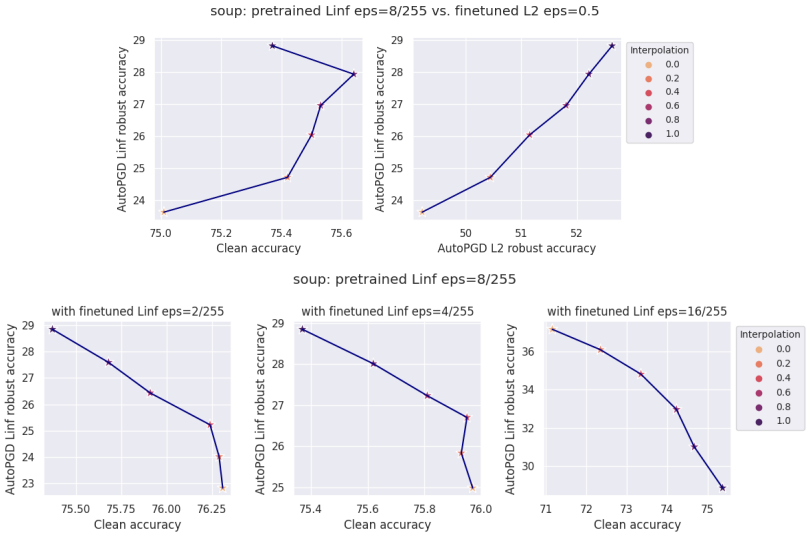
WIDERESNET-28-10 FROM WANG ET AL.

Figure 3: We repeat the ternary plots from Figure 2 for our WideResNets.

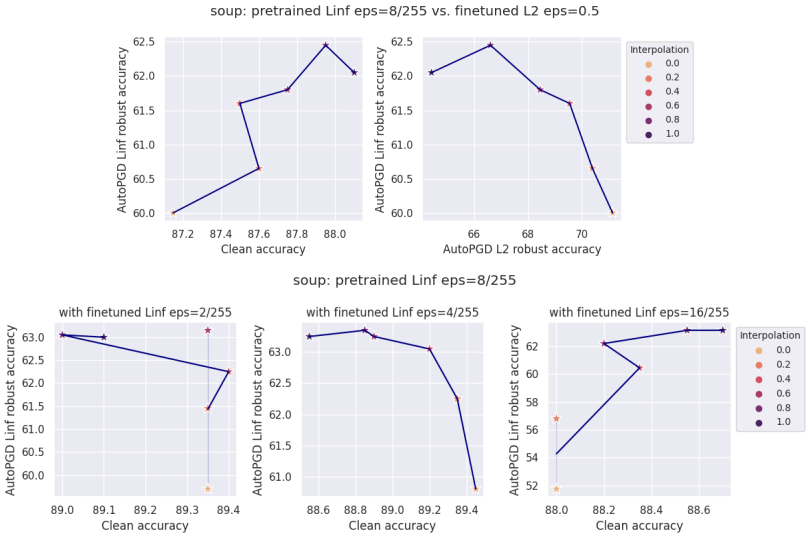
We also perform grid searches with just two ingredient models, similar to the "wise-ft" method of model averaging in [MGJ+22]. We perform wise-ft experiments between the pretrained model (trained on a $L\infty$ $\epsilon = 8/255$ attack in all cases) and each of our finetuned checkpoints. Results are shown in Figure 4. In some experiments, across both the ResNets and WideResNets, averaging a pretrained model smoothly trades off between the clean and robust accuracy. For the ResNet-50 and the Wang et al. WideResNet, we see a slight outward clean-robust accuracy curve in the interpolation path which beats out the straight line connecting the endpoints; this is also seen in the experiments of [MGJ+22], although our experiments don't see absolute improvement in clean or robust accuracy over the endpoints. Additionally, compared to [MGJ+22], our accuracy curves exhibit more irregularities, which is especially apparent in the Gowal et al. WideResNet experiments.

For our wise-ft and ternary grid search experiments, the behavior of clean and robust accuracies vs. averaging is irregular enough such that a comprehensive grid search of the interpolation space is often

RESNET-50



WIDERESNET-28-10 FROM GOWAL ET AL.



WIDERESNET-28-10 FROM WANG ET AL.

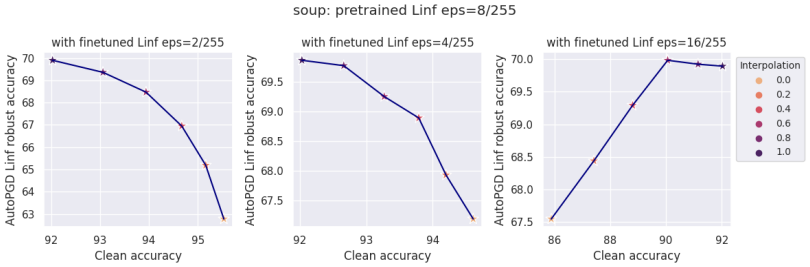


Figure 4: We average between a pretrained and finetuned checkpoint at regular intervals of interpolation coefficients, $(\alpha_1, \alpha_2) = (0,1), (0.2,0.8), (0.4,0.6), (0.6,0.4), (0.8,0.2), (1,0)$.

necessary to find the best performing model. Our RobustBench-sourced WideResNets show non-convexity in the "accuracy landscape" much more consistently than our smaller ResNets, potentially indicating that the "right" soup is harder to find for larger, production-grade models.

Although grid search is tractable when we have two or three ingredients, the size of the search space for a given interpolation step size, such as the 0.2 we used, grows quadratically with the number of fine-tuned models. As such, grid search becomes very computationally expensive for large numbers of soup ingredients. On the contrary, the learned soup remains computationally lightweight for large numbers of ingredients. In training a learned soup, only a set of interpolation coefficients for each layer is trained, and all other parameters are frozen; as such, even for model architectures containing millions of parameters, such as very deep ResNets or CLIPs, the corresponding learned soup only contains trainable parameters in the hundreds or thousands. The main bottleneck with learned soups is memory; training a learned soup requires all model checkpoints to be loaded into a single model. Fortunately, the lightweight nature of learned soup optimization can compensate for this memory cost somewhat. For example, note that the Adam optimizer consumes approximately twice the memory of a model's parameters during training, so a GPU that accommodates training a given model architecture with Adam can also fit a learned soup on three checkpoints of that same model.

4.2 Learned soups

For all the same architectures and model soup configurations we previously explored with grid search, we also train learned soups on a range of AT settings; we use a L_∞ AutoPGD attack for all learned soups and vary the epsilon across $\epsilon = \{0, 2/255, 4/255, 8/255, 16/255\}$, where $\epsilon = 0$ corresponds to clean training. We note that varying the strength of our attack in adversarial training serves as a substitute for the distributional robustness-improving methods in [MGS+22]. In the original model soups investigation, all methods of model averaging were based on optimizing the clean accuracy, and improvements on transfer learning settings simultaneously emerged. When shifting gears from distributional robustness to adversarial robustness, we can no longer optimize learned soups for only clean accuracy, robust accuracy would strictly deteriorate; therefore, it's necessary for us to apply adversarial training on learned soups, but apply training settings that mimic a tradeoff between clean training and adversarial training. Modifying the distortion bound of adversarial training performs this function. (Later, we explore extra experiments where learned soups are trained with TRADES.)

Results for threat model soups and eps soups are shown in Figure 5. The threat model soup for the Gowal et al. WideResNet is our only setting where learned soups strictly underperforms grid search. Otherwise, across all of our models, we see mostly consistent behavior in the clean-robust tradeoff as we vary the strength of our attack in AT of learned soups. Even for our WideResNets, where previous experiments indicate non-convexity in the loss landscape or "accuracy landscape," the accuracy curves on eps soups are mostly convex. In addition, for all eps soups, the learned soup accuracy curves attain better joint clean-robust accuracy than the straight line connecting the two grid search benchmark points, indicating that learned soups trade off clean and robust accuracy along a similar "optimal frontier" to that of grid search.

In some cases, the learned soups method attains a better robust accuracy than was found during grid search. For threat model soups on our ResNets, learned soups traces a suboptimal accuracy curve compared to that implied by grid search; however, the learned soup trained with $\epsilon = 16/255$ attacks obtains the highest robust accuracy out of all soups. For the Gowal et al. WideResNet, all choices of adversarial training ϵ except for $\epsilon = 16/255$ surpass grid search in robust accuracy of the resulting learned soups.

For the ResNet-9, we alternatively train learned soups with TRADES, varying the ratio hyperparameter α as depicted in Equation 2. Our results using TRADES are depicted in Figure 6. The resulting accuracy curves follow similar behavior to those from our experiments with standard AT, but varying the TRADES ratio has a much weaker clean-robust tradeoff effect compared to varying the ϵ in AT.

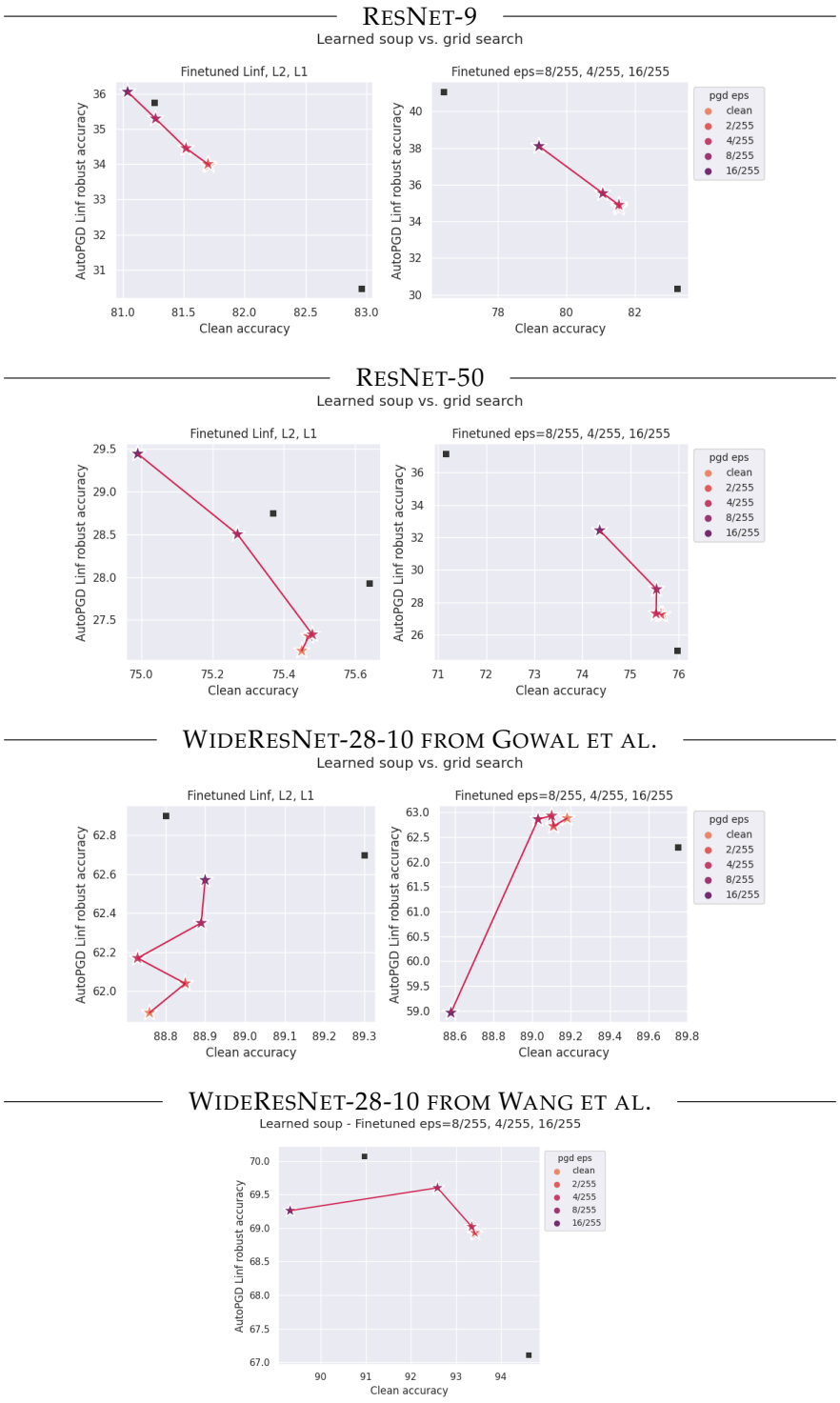


Figure 5: Clean-robust accuracy curves for learned soups as a function of ϵ in adversarial training. We benchmark against the best soups in grid search for the corresponding soup configuration; in each plot, the two points denoted by ■ are the soups with best absolute clean accuracy and best absolute robust accuracy, respectively, found in grid search. Note that for the eps soup on the Gowal et al. WideResNet, these two points actually coincide, so just one appears.

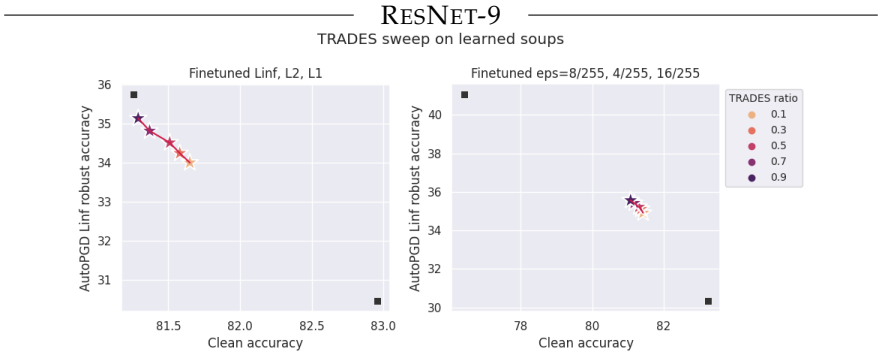


Figure 6: We switch out adversarial training for TRADES in the experiments of Figure 5.

4.3 Robust loss landscapes during model averaging

Our previous experiments strongly hinted at non-convexity in the robust loss landscape of the region between finetuned checkpoints in parameter space. We explicitly verify the behavior of the loss landscape for a model soup between a pretrained checkpoint and two finetuned checkpoints. Via the method in [TPD+18], we find the 2-D plane spanning our three checkpoints in parameter space and evaluate a grid of points on this plane for the robust test loss of the corresponding model. To save computational cost, we evaluate on a random subset of 2000 test samples in CIFAR-10, rather than the full test set of 10000 samples. The robust loss landscapes for our ResNet-9 and the WideResNet from Wang et al. are depicted in Figure 7.

Our robust loss landscapes are a far cry from the convex loss basins found in [JGD+20] [PDT+18] [MGJ+22]. We see significant occurrences of local optima or saddle points in the model soups region of parameter space, even appearing in all of our ResNet-9 contour maps. Such non-convexities are especially frequent for the region between the L_∞ $\epsilon = 8/255$ and L_∞ $\epsilon = 4/255$ checkpoints of the Wang et al. WideResNet. We recall that our grid search on ResNet-9 model soups did not reveal any clear non-convexities in the "accuracy landscape," indicating that model soup grid searches which are too coarse-grained can belie the non-smoothness of model soup behavior in interpolation space.

The brittleness of our robust loss landscapes, even when previous research finds convex clean loss landscapes in model averaging, agrees with conventional wisdom in the adversarial robustness literature about the higher difficulty of robust generalization. [AAL+18] demonstrates that decision boundaries robust to adversarial examples are necessarily more complex than clean decision boundaries on the same data, and [BJH+22] show that robust accuracy is bottlenecked by model capacity.

Empirically, these irregular loss landscapes are evidence that the clean-robust tradeoff for adversarially finetuned models is often not smoothly controlled by grid search. In contrast, according to our results in Figure 5, the learned soups method does consistently control the clean-robust tradeoff even in settings where the underlying loss landscape is highly brittle, such as for the Wang et al. WideResNet. Combined with the computational tractability advantage that learned soups has over grid search between large numbers of finetuned checkpoints, the learned soups method may be the preferred method of model averaging in high-performance adversarial robustness settings.

4.4 Additional Discussion & Limitations

Because there is a (very rough) equivalent between training learned soups and training an adversarially finetuned checkpoint for additional epochs, a significant concern for us is whether learned soups are prone to overfitting. In other words, we may essentially be performing adversarial training on finetuned checkpoints for longer than they "need." To investigate the possibility of overfitting, we train a learned soup with a single set of interpolation coefficients for the total model parameters, rather than distinct coefficients for each layer. Under this constraint, the learned soup optimizes over the same plane in parameter space that grid search evaluates. Across multiple random trials, we find that training consistently converges to a point in interpolation space with weaker clean and robust performance than the best point in grid search, indicating overfitting in this constrained setting.

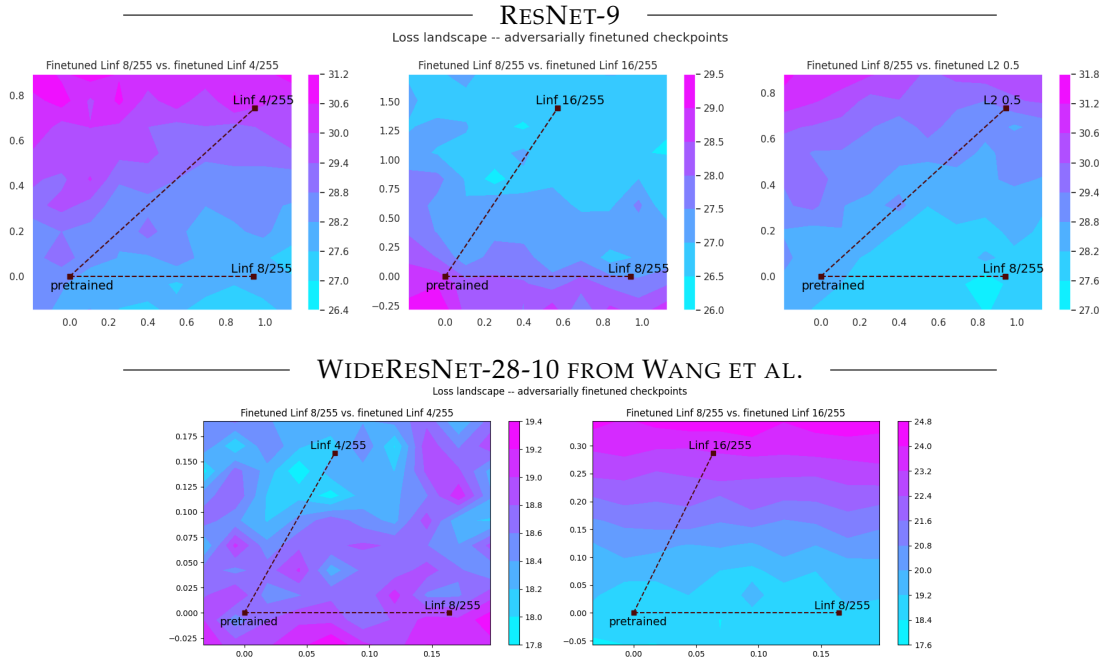


Figure 7: Robust loss landscapes for averaged models between a pretrained checkpoint and two finetuned checkpoints. We choose $\theta(L_\infty, 8/255)$ as the first finetuned checkpoint and choose from $\theta(L_\infty, 4/255)$, $\theta(L_\infty, 16/255)$, or $\theta(L_2, 0.5)$ as the second finetuned checkpoint.

As such, additional methods to mitigate overfitting for learned soups may reap further benefits in learned soup performance. Our detach dropout method performs this function by causing any given interpolation layer to experience only a fraction of the total optimization steps corresponding to the training epochs. Since earlystopping is so effective for avoiding robust overfitting, as shown in [LEJ20], it may be helpful to incorporate earlystopping in learned soups. However, since earlystopping is usually performed with a validation set, splitting a random validation set from the train set for earlystopping learned soups would constitute data contamination when the soup ingredients saw the full train set in pretraining or finetuning. Therefore, in order for earlystopping on learned soups to be viable, earlystopping for the soup ingredients during pretraining and finetuning must also be performed, and all earlystopping throughout the pipeline must use the same exact validation split. This may not be possible for some deployed settings where the pretrained or finetuned checkpoints cannot be trained from scratch.

5 Related Work

Model soups. Averaging neural network parameters has been extensively explored in past deep learning literature. One widely-used method in neural network training recipes, stochastic weight averaging, originates from the insight that averaging an SGD-trained model's parameters with those of earlier checkpoints in the same optimization trajectory leads to a model with better generalization [PDT+18]. Further works have explored averaging between independently finetuned models from the same pretrained checkpoint. [JGD+20] found that, with ResNet-50 models trained on Imagenet, two models that share the beginning of their training trajectory are connected in weight-space by a linear path of equal or better error than either endpoint, a phenomenon dubbed "linear mode connectivity." [BHC20] and [Uma23] study the loss landscape behavior of finetuned models from the same pretrained checkpoint, respectively finding that such finetuned models remain in the same loss basin as that of the pretrained, and that the path between finetuned models in this loss basin often becomes convex very early in pretraining.

In particular, [BHC20] suggests that averaging model parameters confers benefits for transfer learning, a setting explored in subsequent research literature. Indeed, [MGJ+22] performs averaging between

finetuned CLIP models and observes improvements in state-of-the-art performance on both ImageNet and distribution shift tasks. Seperately, [MGS+22] scales up the number of "ingredient" checkpoints used in averaging; they verify high performance of averaged CLIP models for as much as 72 ingredients, validating that larger numbers of finetuned checkpoints can be averaged together for greater returns. To our knowledge, [MGS+22] is also the first work to devise a method of directly learning the averaging coefficients between checkpoint parameters, which they call a "learned soup." Our methods are closely based on the model soup and learned soup of [MGS+22]. Compared to these previous works, which mainly explored model averaging for the original task or transfer learning settings, our work is one of the first to validate model averaging for adversarial robustness; we also extend the "learned soup" method of [MGS+22] by fleshing out a training recipe for model averaging specifically tailored for distributional and adversarial robustness.

Model soups with adversarial training. While the original "model soups" work explored robustness from a distribution shift standpoint rather than an adversarial standpoint, [FSE+23] recently employed model soups for improving the robustness of adversarially trained models. The authors finetuned ResNets and ViTs [ALA+21] [EVP22] for their soups via AT with varying L_p norms ("threat models"), including the L_∞ pgd attack as well as L2 and L1 variants of the AutoAttack introduced in [FM20b]; they find that model soups between these finetuned checkpoints can outperform any single checkpoint in multi-norm adversarial robustness, and that, in the ImageNet setting, these same soups can boost performance on ImageNet distribution shifts. Indeed, we take inspiration from several of the experiments in [FSE+23], especially the manual grid search for soups between exactly three ingredient models. Our work differs from theirs in several aspects:

1. We focus mainly on the standard L_∞ pgd attack as a measure of robustness, and we optimize our soups for simultaneous clean and robust accuracy, rather than multi-norm robustness.

2. We explore model soups between more variations of AT hyperparameters–namely, different epsilons in L_∞ pgd training and intermediate checkpoints of a single AT finetuning run–beyond the different threat models explored in their work.

3. We devote the majority of our attention to the learned soup method, making significant contributions to the training recipe for learned soups and evaluating its potential for robustness settings.

Robust-clean tradeoff. Empirically, a tradeoff between clean and robust accuracy persistently occurs in models trained for adversarial robustness. Many works have analyzed the theoretical underpinnings of this tradeoff or developed adversarial training methods that specially account for this tradeoff. In [DSL+19], the authors observe a decrease in clean accuracy with more epochs or a higher-epsilon pgd attack in adversarial training, and argue that robust and clean models learn fundamentally different features; they construct a classification setting in which any attained classification accuracy for a model confers a corresponding upper bound on the model's robustness.

In the face of this provably extant tradeoff, one popular adversarial training method, TRADES [HYJ+19], decomposes the robust error into the clean error plus a boundary error term for how likely a given data point lies within epsilon-distance of the decision boundary. The authors design an adversarial training procedure around their devised adversarial loss, which comprises the natural error plus a smoothness error representing the "difference" between a data point and its adversarial perturbation. In practice, TRADES can be implemented in any training loop by optimizing on a weighted average of loss on a clean batch and loss on the pgd attack generated from that batch; we use this implementation for our experiments that incorporate TRADES.

Parallel to the observation that adversarial training deteriorates clean accuracy after a certain number of epochs, [LEJ20] also observes robust overfitting, where too many epochs of adversarial training also worsens performance on unseen adversarial examples. The authors employ a validation-based earlystopping procedure to avoid robust overfitting, and show that earlystopping using the baseline AT procedure from[AAL+18] is able to match TRADES in resulting performance.

It is commonly believed in the adversarial robustness literature that robustness requires much more training data compared to clean training. Indeed, recent adversarial training procedures using synthetic

data have made great strides in reducing the robustness-accuracy gap. Works such as [SSO+21] [VST+22] [ZTC+23] have elucidated analytical conditions for synthetic data and pseudo-labels that lead to improved robustness; in particular, [VST+22] proves and experimentally validates that robustness to a synthetic distribution transfers to robustness on the original distribution, devising a FID-like score corresponding to the performance of robustness transfer. Empirically, the adversarial robustness literature has continually obtained state-of-the-art models by employing more powerful generative models (from GANs to DDPM [JAP20] to latent diffusion [RAD+22]) and scaling up the amount of synthetic data to millions of samples.

6 Conclusion

Throughout this work, we explore the potential of model averaging methods such as model soups for managing the clean-robust tradeoff which is ever-present in adversarial robustness settings. Robustness is a more challenging setting, with a more irregular optimization landscape, compared to the clean settings of previous research on model averaging; as such, we refine the learned soups method in order to help find model soups with a desired joint clean-robust performance. We benchmark our learned soup method against grid search for model soups, experimenting on a range of toy models, state-of-the-art robust models, and finetuned checkpoint grids. We find that learned soups can more smoothly control the clean-robust tradeoff compared to grid searches and can obtain higher robust accuracy than grid search in some settings. Our findings show that the *adversarial learned soup* may be a highly effective addition to the arsenal of defenses in the adversarial robustness literature.

References

[AAL+18] Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations (ICLR)* (2018).

[ALA+21] 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: *ICLR* (2021).

[BHC20] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. "What is being transferred in transfer learning?" In: *NeurIPS* (2020).

[BJH+22] Binghui Li, Jikai Jin, Han Zhong, John E. Hopcroft, and Liwei Wang. "Why Robust Generalization in Deep Learning is Difficult: Perspective of Expressive Power". In: *NeurIPS* (2022).

[CJS+19] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. "Adversarial Robustness through Local Linearization". In: *NeurIPS* (2019).

[CWI+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *International Conference on Learning Representations (ICLR)* (2014).

[DSL+19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Mądry. "Robustness May Be at Odds with Accuracy". In: *International Conference on Learning Representations (ICLR)* (2019).

[EVP22] Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. "A Light Recipe to Train Robust Vision Transformers". In: *arXiv preprint* (2022).

[FM20a] Francesco Croce and Matthias Hein. "Minimally distorted adversarial examples with a fast adaptive boundary attack". In: *ICML* (2020).

[FM20b] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *International Conference on Machine Learning (ICML)* (2020).

[FMV+21] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. "RobustBench: a standardized adversarial robustness benchmark". In: *NeurIPS* (2021).

[FSE+23] Francesco Croce, Sylvestre-Alvise Rebuffi, Evan Shelhamer, and Sven Gowal. "Seasoning Model Soups for Robustness to Adversarial and Natural Distribution Shifts". In: *arXiv preprint arXiv:2302.10164* (2023).

[GMS+22] Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. "Patching open-vocabulary models by interpolating weights". In: *NeurIPS* (2022).

[HYJ+19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. "Theoretically Principled Trade-off between Robustness and Accuracy". In: *International Conference on Machine Learning (ICML)* (2019).

[IJC15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *International Conference on Learning Representations (ICLR)* (2015).

[JAP20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *NeurIPS* (2020).

[JGD+20] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. "Linear Mode Connectivity and the Lottery Ticket Hypothesis". In: *International Conference on Machine Learning (ICML)* (2020).

[LEJ20] Leslie Rice, Eric Wong, and J. Zico Colter. "Overfitting in adversarially robust deep learning". In: *NeurIPS* (2020).

[MGJ+22] 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". In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[MGS+22] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *International Conference on Machine Learning (ICML)* (2022).

[ND17] Nicholas Carlini and David Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *IEEE Symposium on Security and Privacy* (2017).

[PDT+18] 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* (2018).

[RAD+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-Resolution Image Synthesis with Latent Diffusion Models". In: *CVPR* (2022).

[SSO+21] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan A. Calian, and Timothy Mann. "Improving Robustness using Generated Data". In: *NeurIPS* (2021).

[TMT+22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. "Elucidating the design space of diffusion-based generative models". In: *NeurIPS* (2022).

[TPD+18] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs". In: *NeurIPS* (2018).

[Uma23] Umarbek Nasimov. "How early can we average neural networks?" MA thesis. MIT, 2023. URL: https://dspace.mit.edu/bitstream/handle/1721.1/151660/nasimov-unasimov-meng-eecs-2023-thesis.pdf.

[VST+22] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. "Robust Learning Meets Generative Models: Can Proxy Distributions Improve Adversarial Robustness?" In: *ICLR* (2022).

[XY10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research* 9 (2010), pp. 249–256.

[YDJ+20] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. "Improving Adversarial Robustness Requires Revisiting Misclassified Examples". In: *ICLR* (2020).

[ZTC+23] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan. "Better Diffusion Models Further Improve Adversarial Training". In: *ICML* (2023).