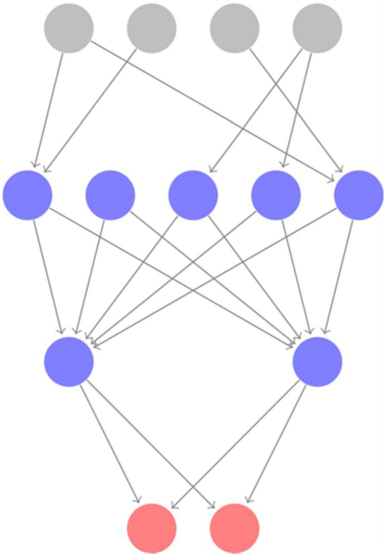
Sparse Model Soups

A Recipe for Improved Pruning via Model Averaging

12th International Conference on Learning Representations (ICLR24)

Max Zimmer

October 2024



ZUSE INSTITUTE BERLIN

Results are joint work of...

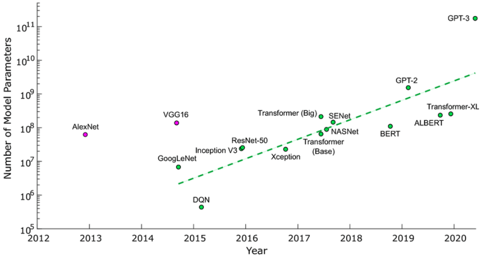
Max Zimmer
Zuse Institute Berlin

Christoph Spiegel
Zuse Institute Berlin

Sebastian Pokutta
Zuse Institute Berlin

Why do we need sparsity?

- Neural Networks are exploding in size
- This yields several problems:
  - **Efficiency:** Long training/inference times
  - **Storage:** Not deployable on phones, ...
  - **Costs:** Costly energy demands
    - Training of Large Language Models can emit as much CO_2 as five cars in their lifetime (Strubell et al., 2019)
    - GPT-3 Training: Estimated cost of 4.6 million USD



Source: Bernstein et al. (2021)

- One potential solution: **Pruning** - The removal of parameters from the network (LeCun et al., 1989; Han et al., 2015).
- Idea: introduce **sparsity** in the parameter tensors to reduce storage- and compute-demands

How to decide what to prune?

Mathematical formulation: $\min_{\mathcal{W}} \mathcal{L}(\mathcal{W}, \mathcal{D})$ s.t. $\|\mathcal{W}\|_0 \leq k$. $\rightarrow$ **Intractable!**

Two different paradigms:

- Regularization: Force parameters towards zero throughout training (e.g., with penalties)
- Saliency criteria: Remove based on a heuristic, e.g., parameter magnitude.

A classical pruning approach:

Iterative Magnitude Pruning (IMP, Han et al., 2015);
Input: A pretrained network θ.
repeat
  PRUNE a fraction of the lowest-magnitude weights;
  RETRAIN the network for a bit;
until *the desired sparsity is reached*;

Leveraging multiple models for better performance

Given m models $\theta_1, \ldots, \theta_m$, can we construct a better model θ?

Ensembles: *Average the outputs* of m models

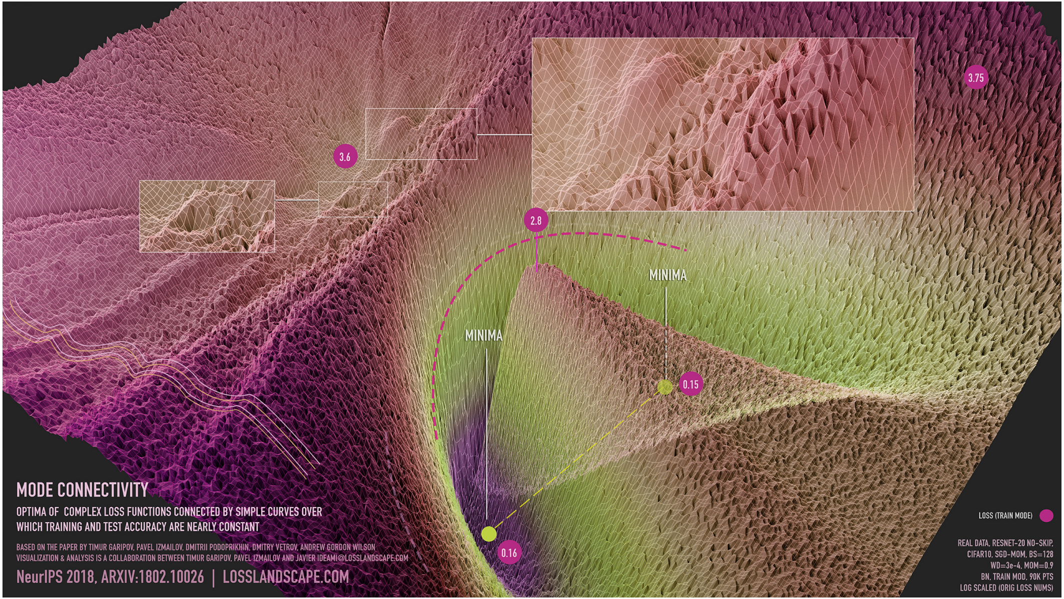
$\rightarrow$ Drastically improves generalization performance

$\rightarrow$ **Problem:** Increases the inference time by a factor of m

Parameter Averaging or Model Soups: *Average the parameters* of m models

$\rightarrow$ New model $\bar{\theta} = \sum_{i \in [m]} \lambda_i \theta_i$ is efficient to use

$\rightarrow$ **Difficulty:** Models θ_i must reside in a linearly connected loss basin. Even averaging models trained with identical initialization but varying seeds degrades performance compared to individual models (Neyshabur et al., 2020).


3.75
3.6
2.8
MINIMA
MINIMA
0.15
0.16
MODE CONNECTIVITY
OPTIMA OF COMPLEX LOSS FUNCTIONS CONNECTED BY SIMPLE CURVES OVER WHICH TRAINING AND TEST ACCURACY ARE NEARLY CONSTANT
BASED ON THE PAPER BY TIMUR GARIPOV, PAVEL IZMAILOV, DMITRII PODOPRIKHIN, DMITRY VETROV, ANDREW GORDON WILSON
VISUALIZATION & ANALYSIS IS A COLLABORATION BETWEEN TIMUR GARIPOV, PAVEL IZMAILOV AND JAVIER IDEAMI@LOSSLANDSCAPE.COM
NeurIPS 2018, ARXIV:1802.10026 | LOSSLANDSCAPE.COM
LOSS (TRAIN MODE)
REAL DATA, RESNET-20 NO-SKIP, CIFAR10, SGD-MOM, BS=128
WD=3e-4, MOM=0.9
BN, TRAIN MOD, 90K PTS
LOG SCALED (ORIG LOSS NUMS)

Motivation

Can we get the benefits of both model averaging and sparsity?

- For ensembles: Easy! Just obtain multiple sparse models and average the outputs!
- But how do we find models that are both sparse and have averageable parameters?

→ We have to resolve two problems! This is the **goal of this work**!

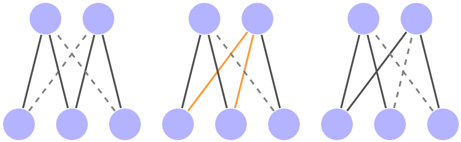
Note: Ensembles should be as diverse as possible, but what about model soups?

→ Model Soup candidates should be diverse enough, but not too diverse?

Problem 1: Averaging destroys sparsity

Averaging sparse models may destroy the sparsity pattern!

θ_1 $\quad\quad$ $\frac{1}{2}(\theta_1 + \theta_2)$ $\quad\quad$ θ_2



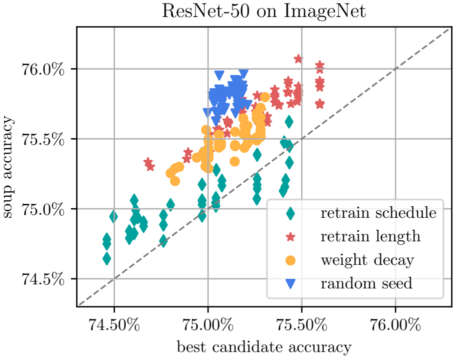
$$\begin{bmatrix} \times & \times & 0 \\ 0 & \times & \times \end{bmatrix} \quad \begin{bmatrix} \times & \times & 0 \\ \times & \times & \times \end{bmatrix} \quad \begin{bmatrix} \times & \times & 0 \\ \times & 0 & \times \end{bmatrix}$$

Problem 2: Finding averageable models

How to obtain models that are averageable?

Idea: Training two models from the same *pretrained* model shouldn't drive them too far apart.

Crucial observation: Pruning a pretrained model and retraining multiple copies with varied hyperparameters (e.g., batch ordering, weight decay) yields averageable models!



(Remark: Hyperparameters should be chosen *reasonably*.)

The recipe

We obtain **sparse models that exhibit superior generalization** performance, **maintaining the sparsity pattern** of their pruned parent in their parameter average.

Idea: Average models after each prune-retrain cycle to ensure identical sparsity!
$\rightarrow$ Proposed algorithm: **Sparse Model Soups (SMS)**

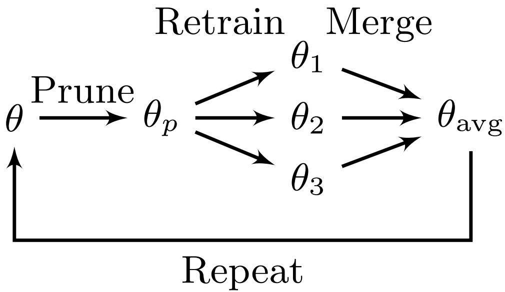


Figure: Sketch for a single phase, $m = 3$.

Comparing SMS against suitable baselines (1)

In each phase, SMS trains m models in parallel for k epochs each.

Suitable baselines:

- IMP: Regular IMP without averaging, i.e., $m = 1$.
- $\text{IMP}_{m\times}$: Extended IMP, where the IMP retraining duration is extended by a factor of m, resulting in $k \cdot m$ retraining epochs per prune-retrain cycle as as many overall epochs as SMS.
- IMP-RePrune: Regular IMP executed m times, averaging performed after the final phase, followed by repruning to address sparsity reduction after averaging.
- Best candidate: Best accuracy among all averaging candidates.
- Mean candidate: Mean accuracy of the averaging candidates.

Comparing SMS against suitable baselines (2)

Table: WideResNet-20 on CIFAR-100 and ResNet-50 on ImageNet: Test accuracies for target sparsities 98% (top) and 90% (bottom) given three prune-retrain cycles.

CIFAR-100 (98%)

	Sparsity 72.8% (Phase 1)			Sparsity 92.6% (Phase 2)			Sparsity 98.0% (Phase 3)		
Accuracy of	$m = 3$	$m = 5$	$m = 10$	$m = 3$	$m = 5$	$m = 10$	$m = 3$	$m = 5$	$m = 10$
SMS	**76.50 ±0.16**	**76.59 ±0.13**	**76.75 ±0.28**	**75.55 ±0.60**	**76.19 ±0.37**	**76.21 ±0.43**	**72.67 ±0.29**	**72.90 ±0.64**	**73.05 ±0.45**
best candidate	75.58 ±0.19	75.71 ±0.08	75.96 ±0.13	74.51 ±0.47	75.01 ±0.74	75.00 ±0.34	71.77 ±0.04	71.77 ±0.37	72.21 ±0.02
mean candidate	75.37 ±0.12	75.58 ±0.03	75.55 ±0.26	74.32 ±0.40	74.71 ±0.48	74.70 ±0.42	71.41 ±0.09	71.61 ±0.40	71.66 ±0.19
$\text{IMP}_{m\times}$	75.85 ±0.26	76.05 ±0.00	75.76 ±0.24	74.09 ±0.24	74.19 ±0.44	74.74 ±0.06	70.92 ±0.07	70.31 ±0.52	71.85 ±0.15
IMP-RePrune		— N/A —			— N/A —		68.19 ±0.44	65.53 ±0.06	63.62 ±0.90
IMP		— 75.54 ±0.41 —			— 74.09 ±0.13 —			— 70.74 ±0.08 —	

ImageNet (90%)

	Sparsity 53.6% (Phase 1)			Sparsity 78.5% (Phase 2)			Sparsity 90.0% (Phase 3)		
Accuracy of	$m = 3$	$m = 5$	$m = 10$	$m = 3$	$m = 5$	$m = 10$	$m = 3$	$m = 5$	$m = 10$
SMS	**76.74 ±0.20**	**76.89 ±0.18**	**77.01 ±0.05**	**76.04 ±0.21**	**76.30 ±0.13**	**76.49 ±0.12**	**74.53 ±0.04**	**74.82 ±0.08**	**74.96 ±0.16**
best candidate	76.07 ±0.01	76.07 ±0.21	76.14 ±0.18	75.48 ±0.16	75.46 ±0.11	75.70 ±0.03	74.00 ±0.03	74.19 ±0.08	74.25 ±0.13
mean candidate	75.99 ±0.04	75.95 ±0.14	75.96 ±0.08	75.40 ±0.11	75.42 ±0.10	75.55 ±0.05	73.94 ±0.03	74.11 ±0.11	74.13 ±0.12
$\text{IMP}_{m\times}$	76.25 ±0.08	76.21 ±0.14	76.46 ±0.04	75.74 ±0.03	75.87 ±0.11	75.93 ±0.03	74.34 ±0.09	74.56 ±0.24	74.50 ±0.09
IMP-RePrune		— N/A —			— N/A —		72.97 ±0.25	72.58 ±0.01	72.08 ±0.12
IMP		— 75.97 ±0.16 —			— 75.19 ±0.14 —			— 73.59 ±0.04 —	

Conclusion

- SMS effectively merges sparse models, maintaining the sparsity pattern and improving generalization and out-of-distribution (OOD) performance.
- Averaging after each prune-retrain cycle and starting from the averaged model significantly enhances the generalization of pruned models.
- SMS consistently outperforms traditional Iterative Magnitude Pruning (IMP) and its extended variants, showing up to 2% improvement in accuracy.
- SMS offers the benefits of parallelization and modularity, enabling practical application in large-scale pruning tasks without increasing inference complexity.

ZUSE INSTITUTE BERLIN

Thank you for your attention!