PopulAtion Parameter Averaging (PAPA)

Alexia Jolicoeur-Martineau, Emy Gervais, Kilian Fatras, Yan Zhang, Simon Lacoste-Julien

May 21, 2023

SAMSUNG
Samsung Advanced
Institute of Technology
AI Lab Montreal

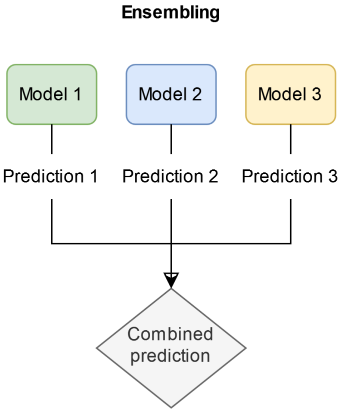
Structure

1. Leveraging a population of models
   - Ensembling
   - Weight averaging
2. PopulAtion Parameter Averaging (PAPA)
   - PAPA-all: occasionally replacing the weights by the average
   - PAPA: slowly pushing the weights toward the average
   - Why does averaging the weights of a population helps?
3. Results
4. Conclusion

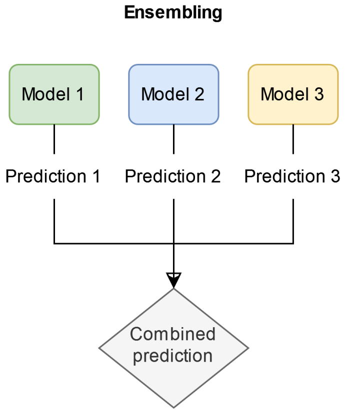
Overview

1. Leveraging a population of models
   - Ensembling
   - Weight averaging
2. PopulAtion Parameter Averaging (PAPA)
   - PAPA-all: occasionally replacing the weights by the average
   - PAPA: slowly pushing the weights toward the average
   - Why does averaging the weights of a population helps?
3. Results
4. Conclusion

The power of ensembling

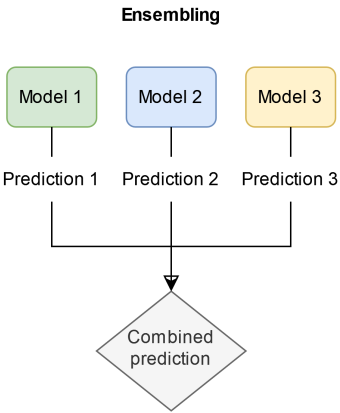


The power of ensembling



Ensembling: Combine the predictions of multiple models for a **massive boost in performance**

The power of ensembling



Ensembling: Combine the predictions of multiple models for a **massive boost in performance**

Simplest form of ensembling: averaging the prediction of all models

The curse of ensembling

Although powerful ensembling has drawbacks:

1. need to train multiple models
2. need to store multiple models
3. need to forward-pass through multiple models at inference-time

The curse of ensembling

Although powerful ensembling has drawbacks:

1. need to train multiple models
2. need to store multiple models
3. need to forward-pass through multiple models at inference-time

Problem: Ensembling is very **expensive** for large models. We do not want to loop through multiple GPT-4 networks...

The curse of ensembling

Although powerful ensembling has drawbacks:

1. need to train multiple models
2. need to store multiple models
3. need to forward-pass through multiple models at inference-time

Problem: Ensembling is very **expensive** for large models. We do not want to loop through multiple GPT-4 networks...

We need a way to leverage a population of models while obtaining a **single** model at the end of training

The curse of ensembling

Although powerful ensembling has drawbacks:

1. need to train multiple models
2. need to store multiple models
3. need to forward-pass through multiple models at inference-time

Problem: Ensembling is very **expensive** for large models. We do not want to loop through multiple GPT-4 networks...

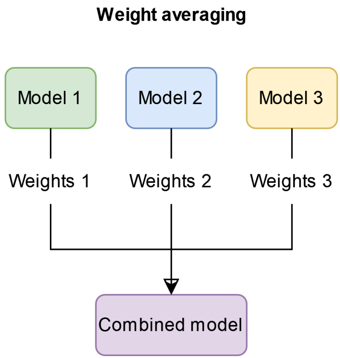
We need a way to leverage a population of models while obtaining a **single** model at the end of training

Solution: weight averaging

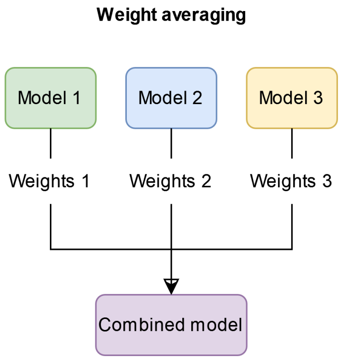
Overview

1. Leveraging a population of models
    - Ensembling
    - Weight averaging
2. PopulAtion Parameter Averaging (PAPA)
    - PAPA-all: occasionally replacing the weights by the average
    - PAPA: slowly pushing the weights toward the average
    - Why does averaging the weights of a population helps?
3. Results
4. Conclusion

Averaging weights as a way to combine models

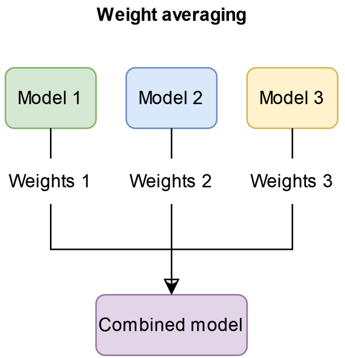


Averaging weights as a way to combine models



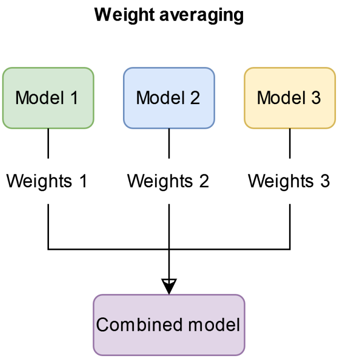
Averaging weights: Combine multiple models into one

Averaging weights as a way to combine models



Averaging weights: Combine multiple models into one
Problem: Averaging does not work well in most scenarios

Averaging weights as a way to combine models



Averaging weights: Combine multiple models into one
Problem: Averaging does not work well in most scenarios
Question: Can we find a way to make it work well?

Permutation-alignment methods for better averaging

Interpolating between weights of two pre-trained networks $\Longrightarrow$ interpolated network has lower train and test loss/accuracy.

Permutation-alignment methods for better averaging

Interpolating between weights of two pre-trained networks $\implies$ interpolated network has lower train and test loss/accuracy.

Permutation alignment minimizes the loss decrease after interpolation.

Permutation-alignment methods for better averaging

Interpolating between weights of two pre-trained networks $\implies$ interpolated network has lower train and test loss/accuracy.

Permutation alignment minimizes the loss decrease after interpolation. How?: By permuting the weights of model B so that the weights (or features) between model A and B are closest.

Permutation-alignment methods for better averaging

Interpolating between weights of two pre-trained networks $\implies$ interpolated network has lower train and test loss/accuracy.

Permutation alignment minimizes the loss decrease after interpolation. How?: By permuting the weights of model B so that the weights (or features) between model A and B are closest.

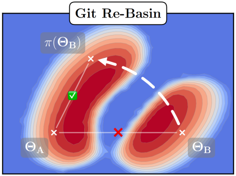


Figure 1: **Git Re-Basin merges models by teleporting solutions into a single basin.** Θ_B is permuted into functionally-equivalent $\pi(\Theta_B)$ so that it lies in the same basin as Θ_A.

Permutation-alignment methods for better averaging

Interpolating between weights of two pre-trained networks $\implies$ interpolated network has lower train and test loss/accuracy.

Permutation alignment minimizes the loss decrease after interpolation. How?: By permuting the weights of model B so that the weights (or features) between model A and B are closest.

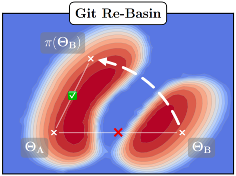


Figure 1: **Git Re-Basin merges models by teleporting solutions into a single basin.** Θ_B is permuted into functionally-equivalent $\pi(\Theta_B)$ so that it lies in the same basin as Θ_A.

Although powerful, permutation-alignment is not enough when averaging ≥ 3 models.

How to ensure that averaging works well?

Averaging work well without permutation-alignment in one scenario [Wortsman et al., 2022]:

How to ensure that averaging works well?

Averaging work well without permutation-alignment in one scenario [Wortsman et al., 2022]:

1. starting from a pre-trained model

How to ensure that averaging works well?

Averaging work well without permutation-alignment in one scenario [Wortsman et al., 2022]:

1. starting from a pre-trained model
2. fine-tune this model on many variations (seed, optimizers, data augmentations, etc.) to produce multiple models

How to ensure that averaging works well?

Averaging work well without permutation-alignment in one scenario [Wortsman et al., 2022]:

1. starting from a pre-trained model
2. fine-tune this model on many variations (seed, optimizers, data augmentations, etc.) to produce multiple models
3. average the weights of a subset of "meaningful" models (model soup)

How to ensure that averaging works well?

Averaging work well without permutation-alignment in one scenario [Wortsman et al., 2022]:

1. starting from a pre-trained model
2. fine-tune this model on many variations (seed, optimizers, data augmentations, etc.) to produce multiple models
3. average the weights of a subset of “meaningful” models (model soup)

Takeaway insight: weight averaging is only beneficial when weights are *similar enough* to average well but *different enough* to benefit from combining them.

How to ensure that averaging works well?

Averaging work well without permutation-alignment in one scenario [Wortsman et al., 2022]:

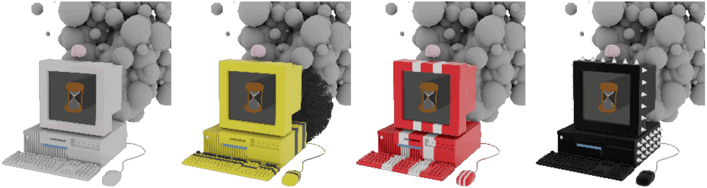
1. starting from a pre-trained model
2. fine-tune this model on many variations (seed, optimizers, data augmentations, etc.) to produce multiple models
3. average the weights of a subset of "meaningful" models (model soup)

Takeaway insight: weight averaging is only beneficial when weights are *similar enough* to average well but *different enough* to benefit from combining them.

This is the key behind PAPA! We ensure during training that weights do not grow too dissimilar over time.

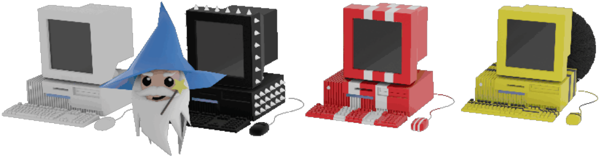
Overview

PAPA-all



4 networks (from 4 different random initializations) are trained independently on different data orderings, regularizations, and data-augmentations; each network learns slightly different features.

PAPA-all

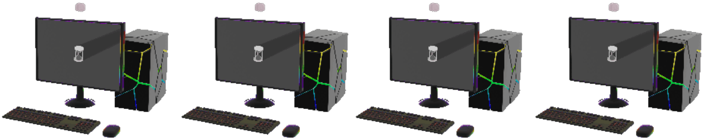


After K epochs, the 4 networks are combined through averaging to create a single averaged network that contains the features of each network and performs significantly better.

PAPA-all

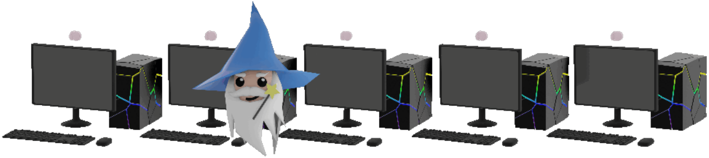
The averaged network is duplicated to form the new population.

PAPA-all



The networks are trained independently again on different data ordering, regularizations, and data-augmentations.

PAPA-all



After k epochs, the networks are averaged again (and the process is repeated every k epochs until the end of training).

PAPA-all

PAPA-all: occasionally replace the weights by the population average of the weights during training

PAPA-all

PAPA-all: occasionally replace the weights by the population average of the weights during training

Phase 1: Parallel SGD training

- Train p models with SGD on the same dataset with variations (random data orderings, regularizations, and data-augmentations)

PAPA-all

PAPA-all: occasionally replace the weights by the population average of the weights during training

Phase 1: Parallel SGD training

- Train p models with SGD on the same dataset with variations (random data orderings, regularizations, and data-augmentations)

Phase 2: Averaging (applied every few epochs)

- Replace every model by the population average of the weights (PAPA-all) or fill the population by averaging from random pair of parents (PAPA-2)
- (Optional) REPAIR the population average of the weights

PAPA-all

PAPA-all: occasionally replace the weights by the population average of the weights during training

Phase 1: Parallel SGD training

- Train p models with SGD on the same dataset with variations (random data orderings, regularizations, and data-augmentations)

Phase 2: Averaging (applied every few epochs)

- Replace every model by the population average of the weights (PAPA-all) or fill the population by averaging from random pair of parents (PAPA-2)
- (Optional) REPAIR the population average of the weights

End:

- Return the model soup (condense the population into a single model)

PAPA-all

PAPA-all: occasionally replace the weights by the population average of the weights during training

Phase 1: Parallel SGD training

- Train p models with SGD on the same dataset with variations (random data orderings, regularizations, and data-augmentations)

Phase 2: Averaging (applied every few epochs)

- Replace every model by the population average of the weights (PAPA-all) or fill the population by averaging from random pair of parents (PAPA-2)
- (Optional) REPAIR the population average of the weights

End:

- Return the model soup (condense the population into a single model)

By averaging occasionally, we gain in performance from averaging, but prevent misalignment due to dissimilar weights!

Model soups

Model soups: [Wortsman et al., 2022] average the weights of multiple chosen models (to combine multiple models into one)

Model soups

Model soups: [Wortsman et al., 2022] average the weights of multiple chosen models (to combine multiple models into one)

Average soup: Population average of the weights

Model soups

Model soups: [Wortsman et al., 2022] average the weights of multiple chosen models (to combine multiple models into one)

Average soup: Population average of the weights

Greedy soup: Greedy average of "meaningful" models

- Sort models in order of decreasing train (or valid) accuracy
- Soup = weights of the first model
- (For loop) Try adding the next model to the soup (averaging all models weights equally); if train accuracy improves, add it to the soup

REnormalizing Permuted Activations for Interpolation Repair (REPAIR)

Repair (REPAIR) [Jordan et al., 2022] mitigate variance collapse after interpolating between two networks through rescaling the preactivations.

REnormalizing Permuted Activations for Interpolation Repair (REPAIR)

Repair (REPAIR) [Jordan et al., 2022] mitigate variance collapse after interpolating between two networks through rescaling the preactivations.

Rescales the convolution/linear layers so that:

$$\mathbb{E}[X_\alpha] = (1-\alpha) \cdot \mathbb{E}[X_1] + \alpha \cdot \mathbb{E}[X_2], \tag{1}$$
$$\mathrm{std}(X_\alpha) = (1-\alpha) \cdot \mathrm{std}(X_1) + \alpha \cdot \mathrm{std}(X_2). \tag{2}$$

where X_α, X_1, X_2 are the features of the interpolated, first, and second networks.

REnormalizing Permuted Activations for Interpolation Repair (REPAIR)

Repair (REPAIR) [Jordan et al., 2022] mitigate variance collapse after interpolating between two networks through rescaling the preactivations.

Rescales the convolution/linear layers so that:

$$\mathbb{E}[X_\alpha] = (1-\alpha) \cdot \mathbb{E}[X_1] + \alpha \cdot \mathbb{E}[X_2], \tag{1}$$
$$\mathrm{std}(X_\alpha) = (1-\alpha) \cdot \mathrm{std}(X_1) + \alpha \cdot \mathrm{std}(X_2). \tag{2}$$

where X_α, X_1, X_2 are the features of the interpolated, first, and second networks.

REPAIR makes the averaged network slightly **more performant** in PAPA

Overview

PAPA: more general algorithm

PAPA-all (low frequency of communications):

1. models trained independently for long periods with no communication
2. rarely replaced by population average

PAPA: more general algorithm

PAPA-all (low frequency of communications):

1. models trained independently for long periods with no communication
2. rarely replaced by population average

PAPA (high frequency of communications):

1. models communicate regularly with the population average
2. slowly push each model toward the population average

PAPA: more general algorithm

PAPA-all (low frequency of communications):

1. models trained independently for long periods with no communication
2. rarely replaced by population average

PAPA (high frequency of communications):

1. models communicate regularly with the population average
2. slowly push each model toward the population average

Solution: At every SGD step, interpolate between current weights and population average:

$$\theta_j \leftarrow \alpha_{papa}\theta_j + (1 - \alpha_{papa})\bar{\theta},$$

where $\alpha_{papa} = 0.999$, θ_j is the weight of the j-th network, and $\bar{\theta}$ is the population average of the weights.

PAPA

Problem: Doing the interpolation at every step slows down training

PAPA

Problem: Doing the interpolation at every step slows down training

Solution: Amortize over a few SGD steps.

PAPA

Problem: Doing the interpolation at every step slows down training

Solution: Amortize over a few SGD steps.

Since $\alpha_{papa} = 0.999^{10} \approx 0.99$, we amortize by doing $\alpha_{papa} = 0.99$ every 10 steps.

PAPA

Problem: Doing the interpolation at every step slows down training

Solution: Amortize over a few SGD steps.

Since $\alpha_{papa} = 0.999^{10} \approx 0.99$, we amortize by doing $\alpha_{papa} = 0.99$ every 10 steps.

When $\alpha_{papa} \approx 0$, we are replacing the weights with the population average. Thus, PAPA-all is a special case of PAPA.

PAPA

Problem: Doing the interpolation at every step slows down training

Solution: Amortize over a few SGD steps.

Since $\alpha_{papa} = 0.999^{10} \approx 0.99$, we amortize by doing $\alpha_{papa} = 0.99$ every 10 steps.

When $\alpha_{papa} \approx 0$, we are replacing the weights with the population average. Thus, PAPA-all is a special case of PAPA.

We can also interpret PAPA as amortizing $\alpha_{papa} = 0.999$ on many steps.
Example: CIFAR-10, averaging every 5 epochs, mini-batch of 64
$\implies \alpha_{papa} = .999^{(781 \text{ iterations} \times 5 \text{ epochs})} \approx 0.02$.

Visualizing the effect of PAPA variants

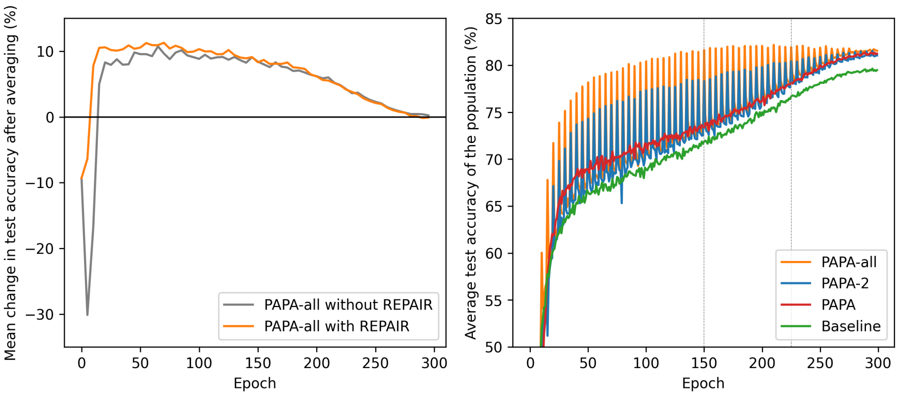


Figure: PAPA variants on CIFAR-100 when averaging every 5 epochs

Overview

Why does averaging the weights of a population helps?

Averaging combines the features from all networks, allowing each network to learn unrealized features discovered by the other networks

Why does averaging the weights of a population helps?

Averaging combines the features from all networks, allowing each network to learn unrealized features discovered by the other networks

We demonstrate that:

- PAPA models share most of their features (proving feature mixing)
- Performance gain immediately after averaging (proving a benefit from feature mixing)

Why does averaging the weights of a population helps?

Averaging combines the features from all networks, allowing each network to learn unrealized features discovered by the other networks

We demonstrate that:

- PAPA models share most of their features (proving feature mixing)
- Performance gain immediately after averaging (proving a benefit from feature mixing)

Thus, surprisingly, averaging weights also leads to averaging features in deep neural networks. More evidence exists in the permutation-alignment literature [Ainsworth et al., 2022].

Overview

1. Leveraging a population of models
   - Ensembling
   - Weight averaging
2. PopulAtion Parameter Averaging (PAPA)
   - PAPA-all: occasionally replacing the weights by the average
   - PAPA: slowly pushing the weights toward the average
   - Why does averaging the weights of a population helps?
3. Results
4. Conclusion

Results

Results

Table 1: Test accuracy from ensembles and soups with varying data augmentations and regularization

	Baseline		PAPA		PAPA-all		PAPA-2	
	Ensemble	GreedySoup	Ensemble	AvgSoup	Ensemble	AvgSoup	Ensemble	AvgSoup
# of models	p	1	p	1	p	1	p	1
CIFAR-10 ($n_{epochs} = 300, p = 10$)								
VGG-11	**95.2** (0.1)	94.0 (0.1)	94.9 (0.1)	**94.8** (0.0)	94.1 (0.2)	94.1 (0.2)	94.5 (0.1)	94.4 (0.1)
ResNet-18	**97.5** (0.0)	96.8 (0.2)	97.4 (0.1)	**97.4** (0.1)	97.3 (0.1)	97.3 (0.1)	97.1 (0.0)	97.1 (0.1)
CIFAR-100 ($n_{epochs} = 300, p = 10$)								
VGG-16	**82.2** (0.1)	77.8 (0.1)	79.6 (0.4)	**79.4** (0.3)	79.0 (0.4)	78.9 (0.4)	79.0 (0.3)	78.9 (0.3)
ResNet-18	**84.3** (0.3)	80.2 (0.6)	82.2 (0.1)	**82.1** (0.2)	81.8 (0.0)	81.8 (0.0)	81.3 (0.3)	81.2 (0.3)
Imagenet ($n_{epochs} = 90, p = 3$)								
ResNet-50	**78.7**	76.8	78.4	**78.4**	77.7	77.7	77.8	77.8
Fine-tuning on CIFAR-100 ($n_{epochs} = 50$, $p = 2, 4, 5$ respectively for EfficientNetV2, EVA-02, ConViT)								
EffNetV2-S	**91.7** (0.3)	91.3 (0.4)	91.6 (0.3)	**91.4** (0.5)	91.4 (0.4)	91.1 (0.4)	91.3 (0.6)	91.3 (0.6)
EVA-02-Ti	90.6 (0.1)	90.4 (0.1)	**90.7** (0.3)	90.6 (0.2)	**90.7** (0.6)	**90.7** (0.5)	90.5 (0.3)	90.4 (0.3)
ConViT-Ti	**88.8** (0.2)	87.9 (0.2)	88.6 (0.2)	**88.4** (0.2)	88.2 (0.2)	88.1 (0.1)	88.2 (0.2)	88.2 (0.3)

Comparing PAPA to single models trained longer

With a single GPU (no parallelization possible), one could either:

1. train PAPA with p networks (in a for-loop) for k epochs
2. train a single model for kp epochs

Comparing PAPA to single models trained longer

With a single GPU (no parallelization possible), one could either:

1. train PAPA with p networks (in a for-loop) for k epochs
2. train a single model for kp epochs

Question: Which choice is best?

Comparing PAPA to single models trained longer

With a single GPU (no parallelization possible), one could either:

1. train PAPA with p networks (in a for-loop) for k epochs
2. train a single model for kp epochs

Question: Which choice is best?

Baseline	PAPA	PAPA-all	PAPA-2
Mean	AvgSoup	AvgSoup	AvgSoup
VGG-16: No data augmentations or regularization			
74.15 (0.1)	**76.04**	75.13	75.10
VGG-16: With random data augmentations			
77.44 (0.1)	**79.36** (0.3)	78.89 (0.4)	78.91 (0.3)
ResNet-18: No data augmentations or regularization			
78.23 (0.6)	78.11	**78.59**	77.90
ResNet-18: With random data augmentations			
79.88 (0.5)	82.06 (0.2)	81.77 (0.0)	81.23 (0.3)

Comparing PAPA to single models trained longer

With a single GPU (no parallelization possible), one could either:

1. train PAPA with p networks (in a for-loop) for k epochs
2. train a single model for kp epochs

Question: Which choice is best?

Baseline	PAPA	PAPA-all	PAPA-2
Mean	AvgSoup	AvgSoup	AvgSoup
VGG-16: No data augmentations or regularization			
74.15 (0.1)	**76.04**	75.13	75.10
VGG-16: With random data augmentations			
77.44 (0.1)	**79.36** (0.3)	78.89 (0.4)	78.91 (0.3)
ResNet-18: No data augmentations or regularization			
78.23 (0.6)	78.11	**78.59**	77.90
ResNet-18: With random data augmentations			
79.88 (0.5)	82.06 (0.2)	81.77 (0.0)	81.23 (0.3)

Observation: training p models with PAPA is often better

Comparing PAPA to single models trained longer

With a single GPU (no parallelization possible), one could either:

1. train PAPA with p networks (in a for-loop) for k epochs
2. train a single model for kp epochs

Question: Which choice is best?

Baseline	PAPA	PAPA-all	PAPA-2
Mean	AvgSoup	AvgSoup	AvgSoup
VGG-16: No data augmentations or regularization			
74.15 (0.1)	**76.04**	75.13	75.10
VGG-16: With random data augmentations			
77.44 (0.1)	**79.36** (0.3)	78.89 (0.4)	78.91 (0.3)
ResNet-18: No data augmentations or regularization			
78.23 (0.6)	78.11	**78.59**	77.90
ResNet-18: With random data augmentations			
79.88 (0.5)	82.06 (0.2)	81.77 (0.0)	81.23 (0.3)

Observation: training p models with PAPA is often better

Conclusion: PAPA is an **efficient way of parallelizing training length** over multiple networks

Overview

Conclusion

The population average of the weights is powerful because contains a **mixture of each network's features**.

Conclusion

The population average of the weights is powerful because contains a **mixture of each network's features**.

When training a population of models, generalization is improved by either:

1. *infrequently* **replacing** the weights with the population average
2. *frequently* **pushing** the networks slightly toward the population average.

Conclusion

The population average of the weights is powerful because contains a **mixture of each network's features**.

When training a population of models, generalization is improved by either:

1. *infrequently* **replacing** the weights with the population average
2. *frequently* **pushing** the networks slightly toward the population average.

Future directions:

1. PAPA for large generative and language models
2. PAPA for continual learning
3. Better understand features propagation between networks in PAPA
4. Different weighting schemes to emphasize generalizable features

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403*, 2022.

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 Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/wortsman22a.html`.