Learning to Compose SuperWeights for Neural Parameter Allocation Search Supplementary

Piotr Teterwak*
Boston University
piotrt@bu.edu

Soren Nelson*†
Physical Sciences Inc
snelson@psicorp.com

Nikoli Dryden
ETH Zürich
nikoli.dryden@inf.ethz.ch

Dina Bashkirova
Boston University
dbash@bu.edu

Kate Saenko
Boston University
saenko@bu.edu

Bryan A. Plummer
Boston University
bplum@bu.edu

1. Implementation Details

We provide the training details of both CIFAR [8] and ImageNet [1] experiments below.

CIFAR. For CIFAR-10 and CIFAR-100 experiments, we train for 200 epochs with an initial learning rate of 0.1. We use SGD with a Nestorov momentum value of 0.9. We use a weight decay value of $5e{-}4$ on all parameters except weight coefficients. We decay the learning rate by a factor of 0.2 at 60, 120, and 180 epochs. We use a batch size of 128, and use asynchronous BatchNorm across two devices (so BatchNorm batch size is 64). We pad images by 4 pixels and crop to 32 x 32 pixels, and also randomly flip and normalize such that it is zero-mean for training. We use the WideResNet 28-10 [17] architecture for all CIFAR experiments.

ImageNet. For ImageNet NPAS experiments we use a WideResNet 50-2 [17] architecture. We train for 90 epochs with a learning rate of 1.6. We use SGD with non-Nestorov momentum value of 0.9. We use a batch size of 1024. We decay the learning rate at epochs 30, 60, and 80 by a factor of 0.1. We use a label smoothing value of 0.1, and a weight decay of 0.0001. The weight decay is not applied to batch norm parameters. We do a linear warmup for the first 10 epochs. We use standard Inception-style data augmentation. For ensembleing experiments we use 64 16GB NVIDIA V100 GPUs, whereas for NPAS experiments we use 4 48GB NVIDIA A40 GPUs.

1.1. Efficient Ensemble Experiments

Homogeneous Ensembles: We use WRN-28-10 models for all of our homogeneous ensembling experiments. For these models, we found that each layer having it's own SuperWeight works well, so did not perform the refinement step.

Heterogeneous Ensembles: We train all models using the settings above except we train on a single GPU, the learning rate is decayed at epochs 60, 120, and 160, and we apply cutout [2] during training (unlike efficient ensembling using homogeneous ensembles, which does not use cutout for a fair comparison to prior work). SWN-Multi-Width is a 3 member WideResNet (WRN) 28-[7,4,3] ensemble. SWN-Multi-Depth/Width is a 4 member WRN 28-[7,4] 16-[7,4] ensemble. All models start from 4 initial depth-binned SuperWeight Clusters and are refined using the gradient similarity threshold from Eq. (3), $\tau = 0.1$. When learning where to share coefficients, SWE-Multi-Width is trained with the gradient similarity threshold from Eq. (2), $\beta = 0.9$ for CIFAR-100 and $\beta = 0.95$ for CIFAR-10. SWN-Multi-Depth/Width uses $\beta = 0.9$ for CIFAR-100 and $\beta = 0.5$ for CIFAR-10. See Section 3.5 for a sensitivity study for hyperparameters β and τ.

1.1.1 Baselines

ShapeShifter Networks [10], Slimmable Networks [16], Universally Slimmable Networks [15] are each trained using the same training settings as the SWE-Multi-Width and SWE-Multi-Depth/Width models (including using cutout for data augmentation).

Slimmable Networks train a Multi-Width network by executing a subset of the channels for a predefined set of width configurations, called switches.

Universally Slimmable Networks extend Slimmable Networks to execute any width of a network within a max and minimum by training at each iteration a random subset of switches and calibrating the the batch normalization statistics of the final switches following training. We train Universally Slimmable using a max width of 7, a minimum width of 3, and the number of widths trained at each iteration, $n = 4$.

*Equal Contribution
†Work done while at Boston University

We calculate batch normalization statistics of network widths [3,4,5,6,7] over one epoch following training.

ShapeShifter Networks (SSNs) automatically learn where to share parameters within a network through clustering coefficients α to form groups and then within each group generating weights for layers from parameter banks via template mixing methods. For SSNs we give each ensemble member its own independent coefficients α and batch normalization parameters. For the Multi-Depth/Width network the layers are grouped manually by depth.

Multi-Scale Dense Network (MSDN) [6] is a Multi-Depth network that provides dense connectivity between early exits and operates at multiple scales for an efficient Anytime Inference model. We train MSDNet using the training scheme from the original paper. All model hyperparameters are kept the same as the original except the network is built with 16 blocks with 12, 24, and 48 output channels for each of the three scales, respectively, in order to match the inference time with other methods.

Hierarchical Neural Ensemble [11] is a Multi-Depth network that trains a tree based ensemble using a novel distillation loss. Results are pulled directly from the original paper with models evaluated for inference time using the author's code. The model uses a modified ResNet-50 with $N = 16$ ensemble members.

Efficient Homogeneous Ensemble Baselines: BatchEnsemble [12] and MIMO [4] are two homogeneous ensembling methods (which we apply to the anytime inference task in Figure 1 of the main paper). BatchEnsemble perturbs weights with rank-1 matrix for each ensemble member. MIMO hard shares all parameters between ensemble members, except for the input convolutional layer and output layer. BatchEnsemble is ineffective as an anytime inference method because there are only N evaluation speeds, where N is the number of ensemble members. MIMO only provides a single inference speed because of its shared backbone.

2. Priority-Queue Weight Template Assignment

In Section 2.2 of our main paper we introduce a method that uses gradient similarity between shared parameters in order to determine where sharing is effective. The intuition behind our approach is that sharing between layers is likely less effective when those layers provide conflicting gradients to the shared parameters. We proposed a greedy approach that would train a model for N epochs and then measure similarity between gradients supplied to the shared parameters aggregated over an entire epoch. Gradient similarity between layers ℓ_i, ℓ_j, which we denote as $\psi_{i,j}$, is computed using Eq. (2) when learning where Weight Templates should share coefficients, and Eq. (3) when learning SuperWeight Clusters. This similarity would be used to construct a priority-queue

Algorithm 1 Priority-Queue Group Assignment

Input: threshold ϵ, priority-queue $Q = \{\text{gradient similarity } \psi_{i,j} : (\text{layers } \ell_i, \ell_j)\}$

```
Groups G = list()
while |Q| > 0 do
    ψ_{i,j}, ℓ_i, ℓ_j = Q.pop()
    if ψ_{i,j} > ε then
        if ℓ_i ∈ g & ℓ_j ∈ g', where g, g' ∈ G then
            G.append(g ∪ g')    ▷ Merge into new group
            delete g, g' from G
        else if ℓ_i ∈ g then    ▷ One layer already belongs to a group
            g = g ∪ {ℓ_j}
        else if ℓ_j ∈ g' then
            g' = g' ∪ {ℓ_i}
        else                    ▷ Neither layer belongs to a group
            G.append({ℓ_i, ℓ_j})
        end if
    else                        ▷ similarity is below threshold, so add any remaining ungrouped layers
        if ℓ_i ∉ g, ∀g ∈ G then
            G.append({ℓ_i})
        end if
        if ℓ_j ∉ g', ∀g' ∈ G then
            G.append({ℓ_j})
        end if
    end if
end while

return G
```

Q, which we traverse merging any layers into a group G that are above some threshold ϵ, with any remaining layers being placed into the same group (see Algorithm 1). Note that in our experiments ϵ is a threshold on gradient similarity, but one could also use a threshold on the number groups instead (or in combination), which we leave for future work. Layers in the same group would continue to share parameters, whereas layers in different groups would no longer completely share parameters.

3. Additional Efficient Ensembling Experimental Results

3.1. Additional Efficient Ensemble Results for Anytime Inference

Prior work in efficient ensembling (*e.g.*, [9, 12, 13]) uses hand-crafted strategies that required ensemble members to have identical architectures and adds diversity by perturbing weights and/or features. In contrast, our SuperWeight Networks, which learn effective soft parameter sharing be-

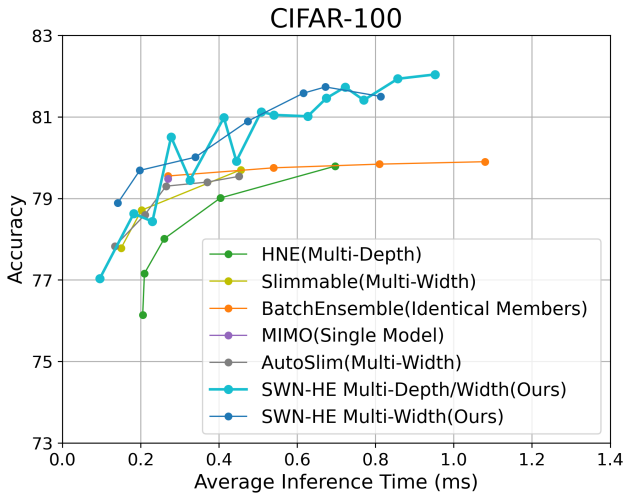

Figure 1. SuperWeight Networkslearns effective soft parameter sharing between members, even for diverse architectures. This enables our approach to support a range of inference times while outperforming prior work in efficient ensembling and anytime inference [4, 11, 12, 14, 16] on CIFAR-100 using WRN-28-5 [17]. Additionally, some efficient ensembling methods like MIMO [4] do not enable multiple inference times at all, represented as a point on the figure.

tween members, even for diverse architectures. As shown in Figure 1, this enables our approach to support a range of inference times while outperforming prior work in efficient ensembling and anytime inference [4, 11, 12, 14, 16] on CIFAR-100 using WRN-28-5 [17]. Additionally, some efficient ensembling methods like MIMO [4] do not enable multiple inference times at all.

3.2. Slimmable Networks and Universally Slimmable Networks Ensemble Comparison

Slimmable Networks [16] and Universally Slimmable Networks (US) [15] both train a dynamic width network such that given a budget at inference time, one can match the budget by running a slim version of the network through executing a subset of the channels at each layer. These methods can be run as an ensemble similar to ours by running inference through multiple widths. In Figure 2 we compare our SWN-Multi-Width model to Slimmable and Universally Slimmable ensembles. SuperWeight Ensembles benefit from ensembling diverse architectures, improving performance. In contrast, ensembling multiple widths of Slimmable and Universally Slimmable Networks only leads to a slight boost, or even decrease, in accuracy.

Method	Top-1 Acc	NLL	ECE
BatchEnsembles [4]	**77.5**	1.02	12.9
MIMO [4]	76.6	0.927	11.2
SuperWeight Networks(**ours**)	76.01	0.885	10.2
SuperWeight Networks(**ours**)	76.6	**0.872**	**8.8**

Table 1. Efficient ensembling comparison on CIFAR-10-C

3.3. CIFAR-10-C

We present results on CIFAR-10-C below. Compared to efficient ensembling baselines, our SWN-HE outperforms others on two out of three metrics, while also providing flexibility not present in prior work via our heterogeneous ensembles and the ability to adjust the number of parameters in our network without changing architecture.

3.4. Ensemble Diversity Analysis

A key attribute of ensembles which makes them effective is their diversity; if the errors of ensemble members are not decorrelated then there is no additional benefit of doing inference through additional ensemble members. In this section, we demonstrate that SuperWeight Networks accomplish this.

We use a diversity metric introduced in Fort et al. [3], which measure the fraction of differing predictions by two ensemble members, normalized the by the error of one of them. We present these results in Table 2; we can see that SuperWeight Networks is more diverse than all other shared parameter methods. One interesting finding is that 120M parameter SuperWeight Networks outperform Standard Deep Ensembles, yet have lower diversity. Looking deeper, the individual model accuracies are improved for SuperWeight Networks (average deep 79.9% vs average SuperWeight Networks 80.4%) Therefore, it seems like SuperWeight Networks helps the model generalize better. This could come from the fact that the parameter factorization limits the space of possible weights. This is especially interesting because WRN-28-10 is already a highly optimized model from a hyper-parameter perspective. Of note is that one of the ensemble members from the SuperWeight Networks gives higher performance (80.5% for the best model) than a standard model (80.1%). Therefore one could use SuperWeight Networks to train a highly performant single model, which signifies another advantage our approach has over prior work [4, 12].

To provide another diversity metric, in Figure 3 we interpolate between two WRN-28-10 ensemble members in parameter space to see if the models indeed are in different optimization basins, and report accuracy at each operating point. We accomplish this by interpolating parameters, and leaving Batch Normalization [7] in train mode because accumulated statistics are not meaningful at interpolated points. If the interpolates have high accuracy, this indicates the en-

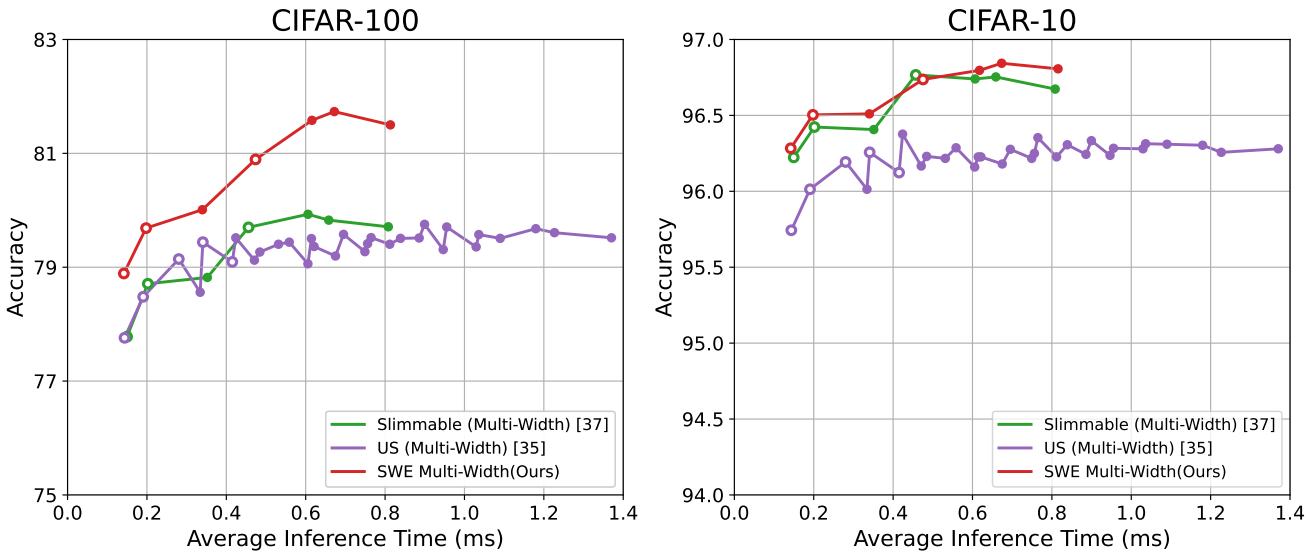

Figure 2. **Slimmable Networks and Universally Slimmable Networks ensemble comparison.** Anytime inference results using Slimmable [16] and Universally Slimmable [15] as ensembles of multiple widths compared to our SWN-Multi-Width.

Method	Params	Diversity
Standard Deep Ensemble [4]	146.0M	0.88
BatchEnsembles [4]	36.5M	0.40
MIMO [4]	36.5M	0.91
SuperWeight Networks (**ours**)	36.5M	0.78
SuperWeight Networks (**ours**)	120.0M	0.85

Table 2. **Diversity on CIFAR-100.** Diversity is measured as the proportion of samples two ensemble members disagree on, normalized by the error rate of one of the member as in [3]. SuperWeight Networks are more diverse than BatchEnsembles, the other shared parameter model. Although our 120M model is slightly less diverse than Standard Deep Ensembles, the performance is higher, due to the average member accuracy being higher than Standard Deep Ensembles

semble members landed in the same optimization basin and therefore are not as diverse as they could be. We find that interpolates have much decreased accuracy compared to the end points, supporting the idea that our model learns diverse ensemble members.

We can see that although all networks experience some degree of accuracy drop between ensemble members, the accuracy drop for the low-parameter model is significantly lower. This seems to indicate that as the function space becomes constrained with a lower number of parameters in the parameter bank, the ensemble diversity starts to suffer.

3.5. Gradient Analysis Hyperparameter Sensitivity

SuperWeight Ensembles learn where to share parameters through two steps of gradient analysis; first to form SuperWeight Clusters (Section 2.2.1), and then to separate SuperWeights by decoupling shared coefficients (Section 2.2.2). Here we explore the sensitivity of the hyperparameters used in each step.

When separating SuperWeights using Eq. (3), we give unique coefficients to SuperWeights where the similarity between the gradients of shared coefficients is less than β. β is used as the threshold ϵ in Algorithm 1 for separating coefficients. In Figure 4(a) we show how sensitive our method is to the selection of β using the SWN-Multi-Width architecture on CIFAR-100 [8]. In addition to reporting results for values of $\beta \in \{0.1, 0.5, 0.9\}$ (note that $\tau = 0.1$), we also report results when no coefficients are shared, and when all coefficients are shared between layers sharing Weight Templates. The results in Figure 4(a) show that no-coefficient and strict coefficient sharing (referred to as "Not Shared" and "Shared," respectively) underperform compared to using our approach. Note that we found optimal values of β to come from dataset-specific tuning.

When generating SuperWeight Clusters using Eq. (2) we split layers into separate groups if the gradient similarity is less than τ. The higher τ, the more groups are formed, the less layers sharing Weight Templates, and the less parameters given to each Weight Template. Setting $\tau \geq 0$ results in sharing between all layers which have gradients that are not conflicting on average. As τ is increased, layer gradients must point in closer directions to be shared. Note τ is given as the input value to ϵ in Algorithm 1. In Figure 4(b) we report best performance comes when $\tau = 0.1$ when using the SWN-Multi-Width architecture on CIFAR-100 [8], which we found to be consistent across settings and datasets. Thus,

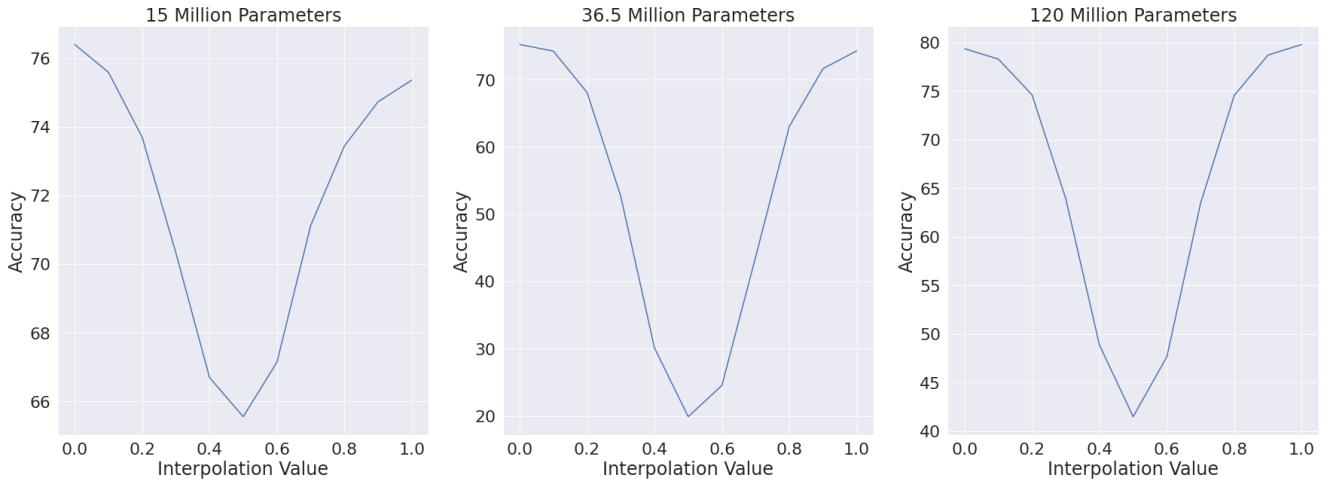

Figure 3. Linear interpolations in parameter space on CIFAR-100, with different numbers of parameters in an ensemble of size 4. We plot accuracy vs interpolation point. Because the accuracy dips, we know the ensemble members are diverse and find different local minima.

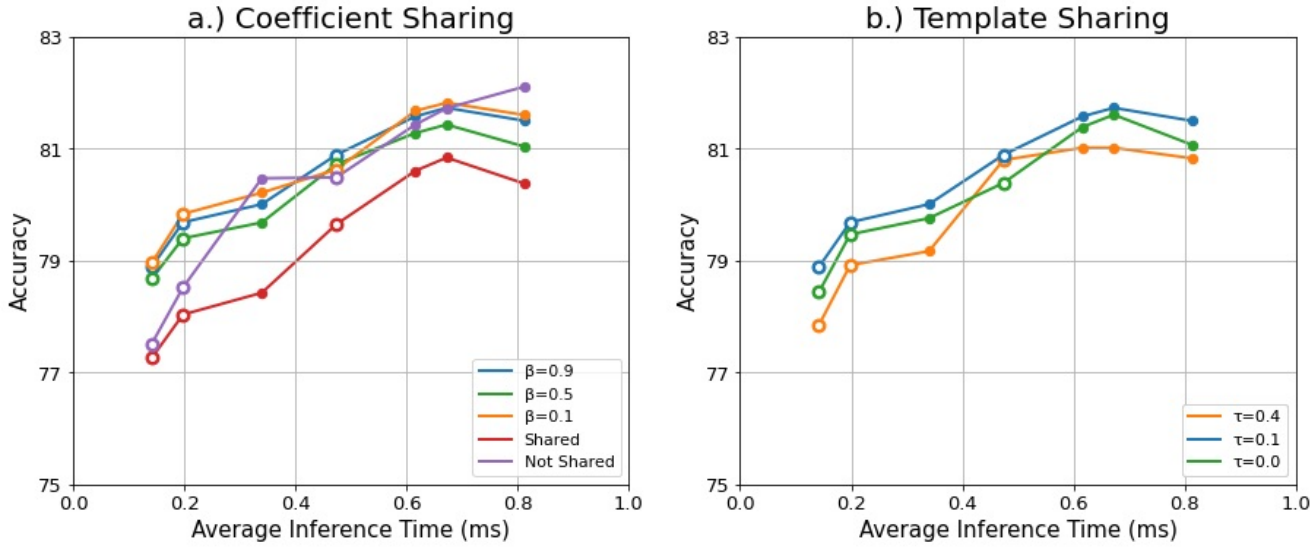

Figure 4. **Gradient analysis hyperparameter sensitivity.** Anytime inference results on CIFAR-100 [8] using SWN-Multi-Width architecture when: **a)** varying the gradient similarity threshold β from Eq. (3) for coefficient sharing, no coefficient sharing, and strict coefficient sharing; and **b)** varying the gradient similarity threshold τ from Eq. (2) for SuperWeight Cluster forming. For both β and τ best performance is between 0.0 and 1.0, indicating that too much sharing and too little sharing are both harmful.

we use this same value of τ across all experiments.

3.6. Performance under severe parameter constraint

One key feature of SuperWeight Networks is that the parameter count is decoupled from backbone. It is therefore interesting to see how the network behaves under stronger parameter constraints. In Table 3, we present CIFAR-100 top-1 accuracies under various parameter constraints. SWN-Single Superweight refers to the homogeneous ensemble which has a single superweight per layer. This is what is presented in Table 2 of our paper. SWN-Gradient Conflict is also a homogeneous ensemble, but with learned SuperWeight Clusters using the gradient conflict criterion. Finally, SWN-Heterogeneous is a WRN 34-8, 28-12, 28-10, and 28-8 ensemble. The gradient conflict criterion becomes more important at lower parameter counts. The improvement of the heterogeneous ensemble over the homogeneous ensemble also increases with decreased parameter counts.

Method	7 million	15 million	36.5 million
SWN-Single Superweight	79.2%	81.3%	82.3%
SWN- Gradient Conflict	80.1%	81.4%	82.2%
SWN- Heterogenous	80.8%	81.6%	82.4%

Table 3. **Changing parameter constraint:** CIFAR-100 Top-1 accuracies. The gradient conflict criterion becomes more important at lower parameter counts. The improvement of the heterogenous ensemble over the homogenous ensemble also increases with decreased parameter counts.

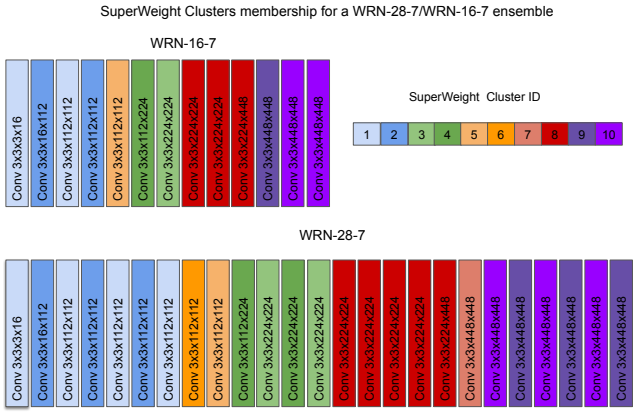

Figure 5. **SuperWeight Cluster Membership** of a two member ensemble, a WRN-28-7 and WRN-16-7. The initial depthwise sharing pattern would segment each network into four equal groups of layers. Our gradient-based learning of SuperWeight clusters allows indvidual layers to specialize. Interestingly, it seems like our method learns to separate adjacent layers into separate SuperWeight clusters. This could increase diversity between ensemble members.

3.7. SuperWeight Cluster Membership

In Figure 5, we present sharing patterns (SuperWeight Cluster membership) of two networks (WRN-28-7/WRN-16-7). The initial depthwise sharing pattern would segment each network into four equal groups of layers. Our gradient-based learning of SuperWeight clusters allows indvidual layers to specialize. Interestingly, it seems like our method learns to separate adjacent layers into separate SuperWeight clusters. This could increase diversity between ensebmle members.

3.8. Diverse Architecture Families

Although we show results primarily on ResNets in our work, our sharing methodology does indeed function well for diverse ensembles. For example, consider an ensemble consisting of a Mobilenetv2 and WRN-28-5, shown in Table 4. We can see that our sharing procedure even helps compared to standard Deep Ensemble performance, despite diverse architectures. Note that these results are over 2 ensemble members, whereas the results in Table 2 of our paper use 4 ensemble members.

Method	Top-1	NLL	ECE
Deep Ensembles	80.1%	0.776	6.5%
SWN-HE	80.3%	0.762	5.3%

Table 4. **Diverse Architecture Families:** We compare the performance of an ensemble consisting of a Mobilenetv2 and WRN-28-5 to standard deep ensembles. We see that our sharing procedure boosts performance despite diverse architectures.

3.9. Efficient Ensembles on CIFAR

In Table 5 we present CIFAR results from the main paper, with error bars (representing standard error) to provide additional context. Note that even with this additional information, it is clear our method outperforms all baselines on CIFAR in the low parameter regime, and even outperforms standard ensembles in the high parameter regime (with 17% fewer parameters). Results we report for baselines are taken from prior work and do not provide error bars.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[3] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[4] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021.

[5] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[6] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[8] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[9] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one:

			CIFAR-100 (clean)			CIFAR-100-C			CIFAR-10		
	Method	Params	Top-1 ↑	NLL ↓	ECE ↓	Top-1 ↑	NLL ↓	ECE ↓	Top-1 ↑	NLL ↓	ECE ↓
(a)	WRN-28-10	36.5M	79.8	0.875	8.6	51.4	2.70	23.9	96.0	0.159	2.3
	BE	36.5M	81.5	0.740	5.6	54.1	2.49	19.1	96.2	0.143	2.1
	BE + EnsBN	36.5M	81.9	n/a	2.8	54.1	n/a	19.1	96.2	n/a	1.8
	MIMO	36.5M	82.0	0.690	2.2	53.7	2.28	12.9	**96.4**	0.123	1.0
	SWN-HO (**Ours**)	36.5M	82.2 ± 0.28	0.702 ± 0.009	2.7 ± 0.04	52.9 ± 0.24	**2.17 ± 0.02**	10.3 ± 0.47	96.3 ± 0.05	0.120 ± 0.002	**0.8 ± 0.08**
	SWN-HE (**Ours**)	36.5M	**82.4 ± 0.02**	**0.663** ±0.001	3.0 ± 0.23	53.0 ± 0.06	**2.17 ± 0.01**	**10.0 ± 0.45**	**96.5 ± 0.02**	**0.115 ± 0.003**	**0.8 ± 0.03**
(b)	Deep Ensembles	146M	82.7	**0.666**	**2.1**	54.1	2.27	13.8	**96.6**	**0.114**	1.0
	SWN-HO (**Ours**)	120M	**82.9 ± 0.05**	**0.666 ± 0.007**	2.2 ± 0.3	**54.7 ± 0.05**	**2.00 ± 0.01**	**10.3 ± 0.01**	**96.6 ± 0.09**	0.119 ± 0.001	**0.8 ± 0.07**

Table 5. **Homogeneous ensembling comparison** on CIFAR-100 (clean) [8] CIFAR-100-C (corrupt) [5], and CIFAR-10 [8] using the WideResNet architecture [17] averaged over three runs. We add error bars representing standard error for our experiments for context. Prior work did not report error bars. **(a)** shows that our approach outperforms prior work in efficient ensembling. **(b)** compares the performance of increasing the number of parameters (without changing the architecture) using our approach compared to standard Deep Ensembles, which trains 4 independent networks as ensemble members.

Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[10] Bryan A. Plummer, Nikoli Dryden, Julius Frost, Torsten Hoefler, and Kate Saenko. Neural parameter allocation search. In *International Conference on Learning Representations (ICLR)*, 2022.

[11] Adria Ruiz and Jakob Verbeek. Anytime inference with distilled hierarchical neural ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[12] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

[13] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33:6514–6527, 2020.

[14] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019.

[15] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019.

[16] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019.

[17] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.