MixtureGrowth: Growing Neural Networks by Recombining Learned Parameters Supplementary

Chau Pham[1*] Piotr Teterwak[1*] Soren Nelson[2*†] Bryan A. Plummer[1]
Boston University[1] Physical Sciences Inc[2]
{chaupham,piotrt,bplum}@bu.edu snelson@psicorp.com

1. Implementation Details

We set up our experiments on image classification tasks where the goal is to recognize the object in an image. This is evaluated using top 1 accuracy, *i.e.*, the percentage of times the model can correctly predict the category of an image. We evaluate our method and the baselines on CIFAR-10 and CIFAR100 [5], which consists of 60K images of 10 and 100 categories respectively, and ImageNet [1], which contains 1.2M images of 1,000 categories.

1.1. CIFAR-10 and CIFAR-100

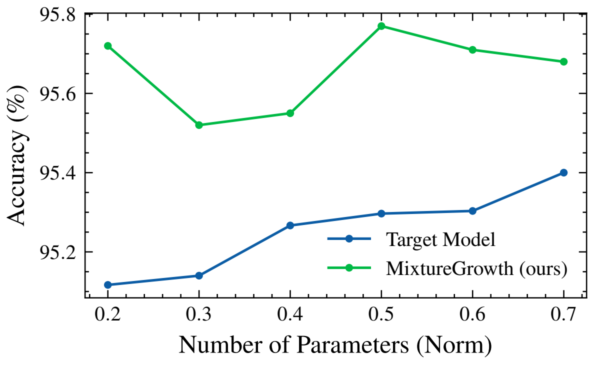
CIFAR-10 and CIFAR100 [5] are composed of 60K images of 10 and 100 categories respectively. In both datasets, we split the data into 50K images for training and 10K images for testing. We perform experiments using Wide Residual Network (WRN) architecture [7], which is a modified version of residual network [3]. We denote a WRN model as WRN-n-k, where n is the total number of convolutional layers, and k is the widening factor. WRN increases the width of each layer by a factor of k while decreasing the depth to improve the performance of the traditional residual network. In this experiment, we choose WRN-28-10 adapted from Savarese *et al.*[1]. The network is trained for 200 epochs in total on a single GPU (Nvidia Titan V 12G) using stochastic gradient descent with momentum 0.9, learning rate 0.1, which is decayed to 0 with a cosine schedule, a weight decay of 5×10^{-4}, and a batch size of 128. For the loss function, we use cross entropy loss. To have a fair comparison, we follow the same setting in prior work [2], where Batch Normalization [4] is only used before each block. Parameters that are specific to our method are set as follows. The total parameter budget for template mixing is set to 36.5M, which is equal to the number of parameters in WRN-28-10 (target network). Note that the parameter budget in our method can be flexible, *i.e.*, it can have a setup where the model has fewer or more parameters thanks to template mixing schemes, see Section 3 for a comparison of varying of parameter budgets. For each layer, the number of templates is set to 2. Except for the first (conv 1) and last (Fully connected) layers, all alpha coefficients are trainable. To train MixtureGrowth, given a trained WRN-28-5 model, we begin with training another WRN-28-5 model for e epochs, where $0 \leq e < 200$ is a hyperparameter. Then, these small models are fused and used to initialize for growing to the full model (*i.e.*, WRN-28-10), which is 4x larger in terms of the number of weights. We train the full model for some more epochs, depending on the FLOPs budget.

It is worth noting that GradMax [2] shrinks the first convolutional layer at every block by 4, resulting in having a small network with wide and narrow layers alternately. In contrast, MixtureGrowth reduces both the widths of its input and output in each layer by 2 to achieve a smaller version of the network. Though the small networks in our method and the baselines are slightly different due to the setting of each method, all of them have the same FLOPs (around 0.25X FLOPs Norm) for a fair comparison.

1.2. ImageNet

ImageNet [1] contains 1,000 categories with 1.2M images for training, 50K for validation, and 100K for testing. We train models using the ResNet-50 architecture [3] for 90 epochs on 4 GPUs (NVIDIA RTX A6000 48G) with a learning rate of 0.1, which is decayed by 0.1 at 30, 60, and 80 epochs with a cosine scheduler. We use stochastic gradient descent with a momentum of 0.9, a batch size of 256, and cross entropy as the loss function. In MixtureGrowth, we share templates between two consecutive layers if they have the same size. The total parameter for template mixing in our method is 25.6M, which is equal to that of a target model. It is worth noting that in GradMax [2], the small network they used required more FLOPs than the one utilized in our method (0.3X FLOPs Norm for GradMax versus 0.26X FLOPs Norm for ours), due to the setup of the authors described in the previous section. However, all other settings remain the same as those mentioned for CI-

*Equal contribution
†Work done while at Boston University
[1]https://github.com/lolemacs/soft-sharing


(a) CIFAR-10

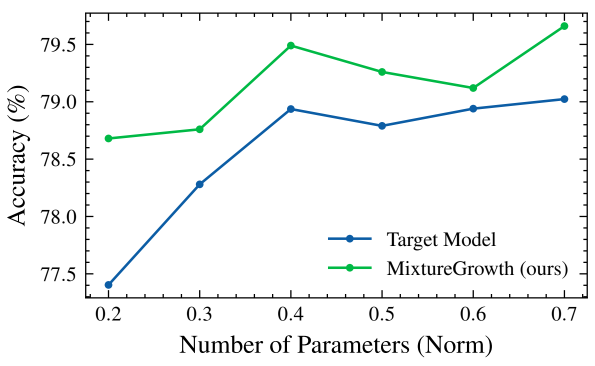

(b) CIFAR-100

Figure 1. Comparison of MixtureGrowth with target model under low parameter settings. Performance measured by top-1 accuracy averaged over 3 runs.

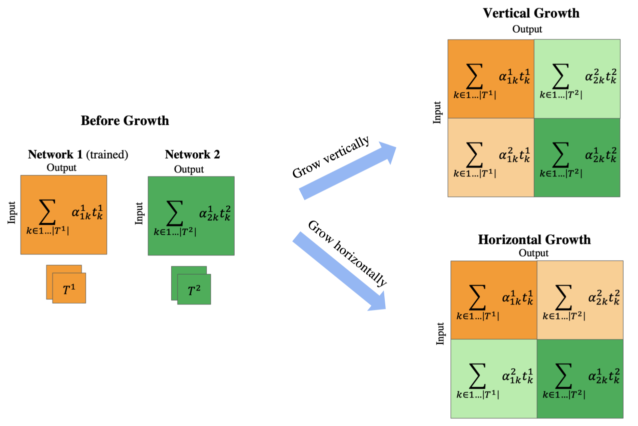


Figure 2. **MixtureGrowth with different growth methods**: Illustration of Vertical growth and Horizontal growth. Quadrants that share templates but have different linear combination coefficients are presented as different shades of the same color.

Table 1. **Network growing comparison on CIFAR-100 using VGG-11** [6] architecture. (a) Performance of baseline models where Firefly is not included due to non-convergence (b) Performance of MixtureGrowth

	Method	Top-1 Acc.	Total FLOPs Norm
(a)	Random	48.52%	1.3X
	GradMax [2]	48.79%	1.3X
(b)	**MixtureGrowth (ours)**	**56.17%**	0.6X

Table 2. Comparison different growth method of MixtureGrowth on CIFAR-100 using WRN-28-10 architecture. We report average accuracy of three runs for each method.

Method	Top-1 Acc.	FLOPs Norm
Horizontal growth	80.66%	0.35X
Vertical growth	**80.82%**	0.35X

FAR above.

To train our method, we start with a trained network whose input and output of each of its layers are half of the sizes of the target network. We train another equal-sized network for e epochs, where $0 \leq e < 90$ is a hyperparameter. Then, these small networks are fused to grow into a full network. We train the fully grown network for some more epochs until run out of the FLOPs budget.

2. Experiments with VGG-11

Besides the WideResnet [7], we conduct experiments on different families of architecture. Table 1 shows the performance of our method and the baselines on CIFAR-100 dataset when growing from 2 small networks. Firefly struggles to converge, resulting in being excluded from the table. However, Random and GradMax give similar results, with about 48% accuracy. Our method performs better than the baselines by a large margin ($\sim$ 7.5%).

3. Low Parameter budgets

Template mixing allows us to share parameters across layers, reducing the number of parameters in the network without the need to change its architecture (such as width and depth). To compare our method with target models under low parameter budgets, we reduce the width of target models so that their number of parameters matches that of our method. Figure 1a illustrates the performance of MixtureGrowth when compared with small target models on CIFAR-10 with the same number of parameters, where MixtureGrowth consistently outperforms the target models under low parameter budgets. We find a similar observation on CIFAR-100 dataset, as shown in 1b. We use WRN architecture [7] for the comparison of both datasets.

4. Horizontal and Vertical Growth

At the growth step, we grow from 2 trained small networks into a large network. Given the trained networks are the 2 diagonal quadrants, there are 2 ways to expand it into 4 quadrants. The first option is Vertical growth, where quadrants that have the same output share the templates (Figure 2, top right). The other way is horizontal growth in which quadrants that have the same input use the same set of templates (Figure 2, bottom right). Table 2 compares the performance of the 2 growth strategies on CIFAR-100 dataset. We notice that Vertical growth slightly outperforms Horizontal growth in terms of performance.

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. 1

[2] Utku Evci, Max Vladymyrov, Thomas Unterthiner, Bart van Merriënboer, and Fabian Pedregosa. Gradmax: Growing neural networks using gradient information. *arXiv preprint arXiv:2201.05125*, 2022. 1, 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 1

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 1

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[7] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016. 1, 2, 3