CL-LSG: Continual Learning via Learnable Sparse Growth

Li Yang
School of ECEE
Arizona State University
lyang166@asu.edu

Sen Lin
School of ECEE
Arizona State University
slin70@asu.edu

Junshan Zhang
Department of ECE
University of California, Davis
jazh@ucdavis.edu

Deliang Fan
School of ECEE
Arizona State University
dfan@asu.edu

Abstract

Continual learning (CL) has been developed to learn new tasks sequentially and perform knowledge transfer from the old tasks to the new ones without forgetting, which is well known as *catastrophic forgetting*. While recent structure-based learning methods show the capability of alleviating the forgetting problem, these methods require a complex learning process to gradually grow-and-prune of a full-size network for each task, which is inefficient. To address this problem and enable efficient network expansion for new tasks, to the best of our knowledge, we are the first to develop a *learnable sparse growth* (LSG) method, which explicitly optimizes the model growth to only select important and necessary channels for growing. Building on the LSG, we then propose *CL-LSG*, a novel end-to-end CL framework to grow the model for each new task dynamically and sparsely. Different from all previous structure-based CL methods that start from and then prune (i.e., two-step) a full-size network, our framework starts from a compact seed network with a much smaller size and grows to the necessary model size (i.e., one-step) for each task, which eliminates the need of additional pruning in previous structure-based growing methods.

1 Introduction

It is well-known that human can learn new tasks without forgetting old ones. However, Deep Neural Networks (DNN) may forget the old knowledge when it is trained to learn new tasks. Such a phenomenon is known as *catastrophic forgetting*. To address this problem and enable to continuously memorize knowledge as human beings do, continual learning (CL) [5], a.k.a, lifelong learning, has recently attracted much attention. It aims to build a model that is incrementally updated over a sequence of tasks, performing knowledge transfer from the old tasks to the new ones without catastrophic forgetting.

In this work, we focus on the structure-based methods [11, 8, 2, 10, 3, 17, 7, 13], which dynamically expand the network capacity to reduce the interference between the new tasks and the previously learned ones. However, such method may incur significant computation costs due to expanding and pruning the full-sized backbone network. For example, DEN [17] and CPG [3] combines the model pruning, weight selection, and model expansion methods, which gradually prune the task-shared weights and then learn additional task-specific weights. The reason why these methods require the additional pruning or searching step is that they grow the network with a redundant size. The redundant growth not only introduces additional computation cost but also interferes with the learning

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

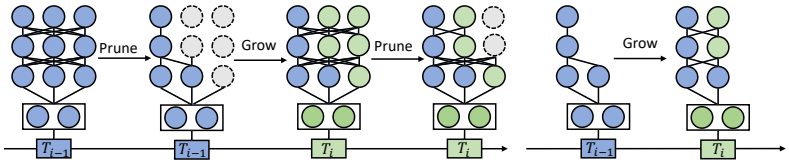

Figure 1: The workflow of a) the conventional structure-based method [17, 3] starts from a full-sized model, and then needs to gradually prune-and-grow for each task; b) our proposed CL-LSG method starts from a small seed model and grows dynamically and sparsely for each task.

performance, calling for a more elegant growing method to achieve efficient CL. Thus inspired, we advocate to *learn a compact and sparse structure for a task in an opposite way by sparsely growing from a small seed network, instead of training and reducing from a large full-sized network.*

To achieve this goal, we first develop a *learnable sparse growth* (LSG) memory mechanism for efficient model expansion, building on which a novel end-to-end structure-based continual learning framework, *CL-LSG*, is devised to *dynamically and sparsely grow the model*. Specifically, different from *all* previous structure-based CL methods that prune a full-sized network, CL-LSG starts from a compact seed network with a much smaller size as shown in Figure 1. As the model weights for old tasks are fixed, we can memorize the required function mappings in a compact model without affecting their accuracy, therefore addressing the catastrophic forgetting issue. By growing from a small network and sparsely expanding the model, the computation cost is significantly reduced, while the learning performance is guaranteed by introducing more degree of freedom for optimization via the expanded network through LSG method. We validate CL-LSG against multiple SOTA methods, which shows 0.71%, 1.0% and 1.2% accuracy improvement on CIFAR-100 Split, CIFAR-100 Superclass and MiniImageNet Split benchmarks, respectively, with the same or even smaller growing model size.

2 Related Work

Continual Learning via Structure-based Method. Structure-based approaches [11, 8, 2, 10, 3, 17, 7, 13] adapt the network architecture with a sequence of tasks. As continually growing the architecture retains the model redundancy, some approaches, e.g., DEN [17] performs model compression before expansion to obtain a more compact model. Newly added weights and old weights are both adapted for the new task with sparse constraints. CPG [3] combines the model pruning, weight selection and model expansion methods, which gradually prunes the task-shared weights and then learns additional task-specific weights.

Table 1: Grow metrics for different methods

Grow metrics	Works
Target accuracy/loss	DEN[17],CPG[3], Lifelong tickets[1]
Grow by default	PNN[11], DAN[10], APD[16], SpaceNet[12], DER[14]

In the structure-based methods, when to grow or if it is necessary to grow for each task is an open question. There are mainly two kinds of grow metrics adopted nowadays in the literature as illustrated in Tab.1: 1) target accuracy/threshold: the current model will grow if the accuracy cannot reach the target accuracy which is obtained by training the full size model from scratch; 2) grow by default: the model will grow directly for each task by default. In this work, we adapt the target accuracy grow metric in the experiments following [17, 3, 1].

Model Growth. Recently, [18] proposes a method to dynamically grow networks for single task by continually re-configuring their architecture during training, which aims to reduce the computational cost. Such a re-configuring process is achieved by learning a channel-wise mask, namely *grown-mask*, $\boldsymbol{m}^g = \{m_n^g\}_{n=1}^N$ for a N-layer convolutional neural network, where each binary element $\boldsymbol{m}_{i,n}^g \in \{0, 1\}$ is associated with a channel i, to enable training-time pruning ($\boldsymbol{m}_{i,n}^g = 1 \rightarrow 0$) and growing ($\boldsymbol{m}_{i,n}^g = 0 \rightarrow 1$) dynamics. The learnable mask variable can be jointly optimized with weights $\boldsymbol{w}$ using data $(\boldsymbol{x}, \boldsymbol{y})$, which is formulated as follows:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{m}^g; \boldsymbol{x}) = \mathcal{L}(f(\{\boldsymbol{w} \odot \boldsymbol{m}^g\}; \boldsymbol{x}), \boldsymbol{y}) + \lambda ||\boldsymbol{m}^g||_0 \tag{1}$$

where $\boldsymbol{w} \odot \boldsymbol{m}^g$ is a general expression of growing channels and $\mathcal{L}$ denotes a loss function (e.g., cross-entropy loss for classification). The l_0 term encourages the sparsity of the grown-mask $\boldsymbol{m}^g$ so as to limit the grow strength, and λ is a coefficient scaling factor.

3 Methodology

3.1 Proposed Learnable Sparse Growth (LSG) with Attentive Sparse Mask

As alluded to earlier, such *two-stage* (i.e., grow-then-prune) conventional structure-based approaches clearly incurs additional computation cost. In this work, we propose a novel approach, namely *learnable sparse growth*, to enable efficient and *sparse* model growth for learning a new task. To address these important issues, we propose a novel *learnable sparse grow* method, through introducing a learnable kernel-wise mask $\boldsymbol{m}^a$ to selectively pick the kernels among all grown channels, named as **attentive-mask**. As shown in Figure 2, the kernel-wise mask $\boldsymbol{m}^a$ introduces sparsity in the grown weights, which can be jointly optimized as:

$$\begin{aligned}&\mathcal{L}(\boldsymbol{w}, \boldsymbol{m}^g, \boldsymbol{m}^a; \boldsymbol{x}) \\ =&\mathcal{L}(f(\{\boldsymbol{w} \odot \boldsymbol{m}^g \odot \boldsymbol{m}^a\}; \boldsymbol{x}), \boldsymbol{y}) + \lambda||\boldsymbol{m}^g||_0 \end{aligned} \tag{2}$$

where $\boldsymbol{w}$ is the full size model. Intuitively, minimizing Equation (2) explicitly optimizes the model growth in a way that only important and necessary channels for learning the current task will be added to the backbone network. Compared with prior growth method [18], which works for single task learning, our contribution lies in being the first to propose an efficient *continual learning* framework with *learnable sparse growth* as the network growing technique. By doing so, our LSG brings multiple benefits in CL setting, including: 1) it mitigates the unstable training on relatively smaller CL new task dataset; 2) it reduces the overfitting issue and achieves better accuracy with the same model size; 3) the sparse weights from attentive mask for current task can be further re-trained for the next task without interfering previous tasks. The detailed explanation of the proposed LSG is relegated to the appendix-A.1.

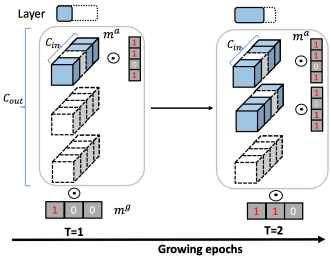

Figure 2: Learnable sparse growth (LSG) with attentive-mask, $\boldsymbol{m}^a$

3.2 Proposed Framework of Continual Learning via Learnable Sparse Growth (CL-LSG)

Clearly, our LSG method can work independently and serve as a general remedy for expanding the network model in a systematic way. Taking advantage of our LSG, we next propose an end-to-end structure-based continual learning framework, *CL-LSG*, with sequentially growing from a small seed network as shown in Figure 3. Next, we present our framework in the sequential-task manner.

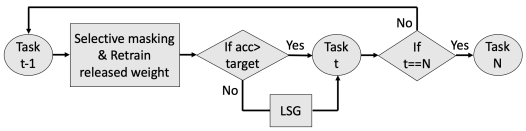

Figure 3: CL-LSG framework.

Learning Task 1: Given the first task, we start from a basic seed network and then gradually grow it by using the proposed learnable sparse growth method in Equation (2). After training, the current model will serve as the backbone model for the next new task.

Learning Tasks 2, ..., N: Assume that in task t, the model that can handle task 1 to $t-1$ has been constructed. We follow a two-stage strategy to learn the model for task t:

stage-1: Pick and reuse. Before deciding to grow the model, we want to fully utilize the current model learned from previous tasks through utilizing two techniques: i) **Selective masking**: we apply a learnable soft kernel-wise mask to the preserved model, so as to select/filter the important weights from previous tasks for current task. ii) **Re-training the released weights**: benefiting from the sparsity generated from our LSG method, the preserved model contains sparse weights that can be re-trained for current task without interfering previous tasks. The detailed explanation of the two techniques is relegated to the appendix.

stage-2: Grow - LSG. After stage-1 training, if the current accuracy is lower than the target accuracy, we will adapt the proposed learnable sparse growth method to integrate more but only necessary features to the current model. Note that, we follow the same target accuracy setup from training full size model as prior works [17, 3, 1], which is also discussed in Table 1.

4 Experiments

We evaluate our CL-LSG on CIFAR-100 Split 10 tasks, CIFAR-100 Superclass 20 tasks and Mini-ImageNet Split 10 tasks. The detailed experimental setting, compared methods and training hyper-parameters are relegated to appendix-A.4.

Table 2: Accuracy on CIFAR-100 Superclass 20 tasks

Methods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	**Avg**	Model Size
Scratch	65.8	**78.4**	76.6	82.4	82.2	84.6	78.6	84.8	83.4	89.4	87.8	80.2	84.4	80.2	52.0	69.4	66.4	70.0	87.2	91.2	78.8	20x
Fine-tune	65.2	76.1	76.1	77.8	85.4	82.5	79.4	82.4	82.0	87.4	87.4	81.5	84.6	80.8	52.0	72.1	68.1	71.9	88.1	91.5	78.6	20x
Grow	67.0	73.8	74.4	75.2	81.4	81.2	78.8	80.4	80.6	85.4	85.8	80.4	81.2	80.6	50.8	68.8	66.4	68.2	84.2	88.4	76.5	1.5x
DEN	66.4	78.0	77.4	78.8	81.6	81.8	76.0	80.4	79.8	85.0	85.2	78.8	83.2	81.6	50.4	71.2	66.8	79.4	85.0	90.2	77.4	2.1x
CPG	67.0	**79.6**	77.2	**82.0**	86.8	**87.2**	82.4	**85.6**	86.4	**89.6**	90.0	84.0	87.2	84.8	55.4	**73.8**	**72.0**	71.6	**89.6**	92.8	81.2	1.5x
Ours	**67.2**	76.8	**79.6**	81.8	**87.4**	86.8	**84.2**	83.8	**87.8**	89.4	**91.0**	**84.6**	**87.2**	**85.0**	**55.4**	73.6	71.2	**73.8**	89.2	**94.6**	**82.2**	1.5x

Table 3: Model size on CIFAR-100 Superclass 20 tasks

Methods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Model Size
Scratch	1x	2x	3x	4x	5x	6x	7x	8x	9x	10x	11x	12x	13x	14x	15x	16x	17x	18x	19x	20x	20x
Fine-tune	1x	2x	3x	4x	5x	6x	7x	8x	9x	10x	11x	12x	13x	14x	15x	16x	17x	18x	19x	20x	20x
Grow	0.3x	0.4x	0.48x	0.56x	0.65x	0.75x	0.75x	0.83x	0.94x	0.94x	1.12x	1.18x	1.24x	1.24x	1.30x	1.30x	1.38x	1.44x	1.44x	1.5x	1.5x
DEN	1.0x	1.06x	1.13x	1.13x	1.19x	1.25x	1.32x	1.32x	1.40x	1.46x	1.46x	1.55x	1.61x	1.61x	1.61x	1.70x	1.82x	1.90x	2.02x	2.1x	2.1x
CPG	1.0x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x
Ours	**0.3x**	**0.4x**	**0.48x**	**0.56x**	**0.65x**	**0.75x**	**0.75x**	**0.83x**	**0.94x**	**0.94x**	**1.12x**	**1.18x**	**1.24x**	**1.24x**	**1.30x**	**1.30x**	**1.38x**	**1.44x**	**1.44x**	1.5x	1.5x

4.1 Quantitative Evaluation

Accuracy comparison. The specific accuracy of each task and final average accuracy are illustrated in Table 2 for CIFAR-100 Sup dataset. The accuracy of training from scratch for each task individually serves as the target accuracy for our method. In practice, if the accuracy for current task after growing still cannot reach the target accuracy, it will directly move to next task. As shown in Table 2, compared with Grow Only methods - PNN and Grow, our method could significantly improve the accuracy for each task, with an even smaller model size. In addition, compared with Grow-and-prune methods - DEN and CPG which require one order more training time, our method achieves better results than DEN on all tasks and CPG on most of tasks. Similar phenomenon can be observed with CIFAR-100 Split and MiniImageNet results as shown in Table 4 and Table 6 in the Appendix. To summarize, our method consistently show clear accuracy gain against SOTA results on three benchmarks with the same or even less model size. Compared with CPG, our method could achieve **0.71**%**, 1.0**% **and 1.2**% accuracy improvement on CIFAR-100 Split, CIFAR-100 Superclass and MiniImageNet Split.

Model size and training time comparison. Table 3 summarize the model size for each task on CIFAR-100 Superclass. The similar table for CIFAR-100 Split and MiniImagnet split are reported in the appendix. Our method could guarantee the smallest model size for each task during training. Furthermore, it is interesting to observe that CL-LSG could achieve good accuracy without growing in certain tasks. For example, CL-LSG does not need to grow the model on $7_{th}, 10_{th}, 14_{th}, 16_{th}$ and 19_{th} tasks. Such phenomenon reveals that the current model trained from previous tasks has strong generality to the current task. Moreover, we measure the **training time** of each method on CIFAR-100 Sup dataset as reported in Figure 4. Note that, the training time represents the whole time of training all tasks. Compared to the CPG and DEN, CL-LSG achieves $\sim 1.4\times$ and $\sim 3.5\times$ speedup with higher accuracy.

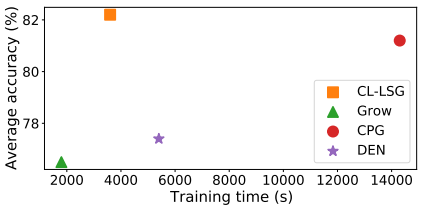

Figure 4: training time and average accuracy comparison between methods

5 Conclusion

In this work, we propose CL-LSG, a novel end-to-end continual learning framework to *dynamically and sparsely grow a model*. Furthermore, to efficiently and appropriately expand network structure for new task, we develop a learnable sparse growth method eliminating the additional pruning step in previous structure-based CL method. We conduct extensive experiments to corroborate the superiority of CL-LSG over current structure-based approaches in multiple CL benchmarks, and characterize the relationship between growth ratio and accuracy for each task.

Acknowledgement. The work of L. Yang and D. Fan was supported in part by the U.S. National Science Foundation Grants No. 1931871 and No. 2144751. The work of S. Lin and J. Zhang was supported in part by the U.S. National Science Foundation Grants CNS-2203239, CNS-2203412, RINGS-2148253, and CCSS-2203238.

References

[1] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2020.

[2] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[3] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] Eric Jang et al. Categorical reparameterization with gumbel-softmax, 2017.

[5] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[7] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR, 2019.

[8] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[9] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[10] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018.

[11] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[12] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Spacenet: Make free space for continual learning. *Neurocomputing*, 439:1–11, 2021.

[13] Tom Veniat, Ludovic Denoyer, and MarcAurelio Ranzato. Efficient continual learning with modular networks and task-driven priors. In *International Conference on Learning Representations*, 2021.

[14] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021.

[15] Li Yang, Zhezhi He, Junshan Zhang, and Deliang Fan. Ksm: Fast multiple task adaption via kernel-wise soft mask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13845–13853, 2021.

[16] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representations*, 2020.

[17] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[18] Xin Yuan, Pedro Henrique Pamplona Savarese, and Michael Maire. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 4
- Did you include the license to the code and datasets? [Yes] The data and code are proprietary
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See appendix
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Will be released if published
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See appendix
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Understanding Learnable Sparse Growth with Attentive Sparse Mask

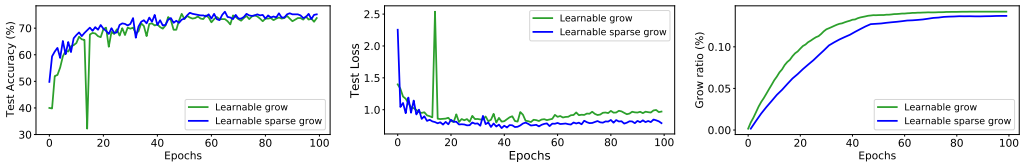

Figure 5: Training curve of learnable growth [18] (green) v.s. our learnable sparse growth (blue) on one sample task of CIFAR-100 Superclass benchmark using VGG16-BN model. Left: test accuracy curve; Middle: test loss curve; Right: grow ratio curve.

To better understand the advantage of introducing the attentive-mask $\boldsymbol{m}^a$ for enabling learnable sparse growth, we further analyze the learning performance of Equation (2) from the viewpoint of optimization, which is summarized in the following Proposition A.1.

Proposition A.1. *Denote* $\mathcal{L}(\boldsymbol{w}_1^*, (\boldsymbol{m}^g)_1^*, (\boldsymbol{m}^a)_1^*) = \min_{\boldsymbol{w},\boldsymbol{m}^g,\boldsymbol{m}^a} \mathcal{L}(\boldsymbol{w}, \boldsymbol{m}^g, \boldsymbol{m}^a; \boldsymbol{x})$ *and* $\mathcal{L}(\boldsymbol{w}_2^*, (\boldsymbol{m}^g)_2^*) = \min_{\boldsymbol{w},\boldsymbol{m}^g} \mathcal{L}(\boldsymbol{w}, \boldsymbol{m}^g; \boldsymbol{x})$. *It can be shown that*

$$\mathcal{L}(\boldsymbol{w}_1^*, (\boldsymbol{m}^g)_1^*, (\boldsymbol{m}^a)_1^*) \leq \mathcal{L}(\boldsymbol{w}_2^*, (\boldsymbol{m}^g)_2^*).$$

Remarks. *Proposition A.1 indicates that the proposed LSG method can always achieve lower loss values compared to the traditional learnable growth.* This performance improvement can be further interpreted from the following perspective: The conventional learnable growth in Equation (1) is mathematically equivalent to the following reformulation by multiplying an identity tensor $\mathbb{1}$:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{m}^g; \boldsymbol{x}) = \mathcal{L}(f(\{\boldsymbol{w} \odot \boldsymbol{m}^g \odot \mathbb{1}\}; \boldsymbol{x}), \boldsymbol{y}) + \lambda ||\boldsymbol{m}^g||_0. \tag{3}$$

That is to say, Equation (1) implicitly puts a constraint on $\boldsymbol{m}^a$ that $\boldsymbol{m}^a = \mathbb{1}$, whereas in Equation (2) the attentive-mask $\boldsymbol{m}^a$ can be optimized over a larger space with more degree of freedom. Thus, to summarize, compared with prior growth method [18], which works for single task learning, our contribution lies in being the first to propose an efficient *continual learning* framework with *learnable sparse growth* as the network growing technique. Moreover, as explained in above technical details, our LSG method mainly incorporates a new attentive sparse mask that has never been used in earlier works, and brings multiple benefits in CL setting, including: 1) it mitigates the unstable training on relatively smaller CL new task dataset; 2) it reduces the overfitting issue and achieves better accuracy with the same model size; 3) the sparse weights from attentive mask for current task can be further re-trained for the next task without interfering previous tasks.

A.2 Selective Masking on Previous Tasks

To mitigate catastrophic forgetting and utilize the useful information from previous tasks, we first freeze the model weights, and only learn a binary mask to select the important weights for current task. Inspired by [15], we apply a learnable kernel-wise mask in our CL-LSG framework, named as **selective-mask** $\boldsymbol{m}^s$, to maximize the utilization of the preserved weights learned from previous tasks, formulated as:

$$\mathcal{L}_t(\boldsymbol{w}_t; \boldsymbol{x}_t) = \mathcal{L}_t(f(\{\boldsymbol{w}_{t-1} \odot \boldsymbol{m}_t^s\}; \boldsymbol{x}_t), \boldsymbol{y}_t) \tag{4}$$

where $\boldsymbol{w}_{t-1}$ denotes the preserved weights for task $t-1$ including all the weight from task 1 to $t-1$. The conventional way [9] of generating the binary trainable mask is to train a learnable real-valued mask ($\boldsymbol{m}^r$) followed by a hard threshold function (e.g., sign function) to binarize it. Different from that, in this work, we adopt a method to better estimate the gradient by using Gumbel-Sigmoid trick. First, we relax the hard threshold function to a continual sigmoid function $\sigma(\boldsymbol{m}^r)$. Then, to learn the binary mask, we leverage the Gumbel-Sigmoid trick, inspired by Gumbel-Softmax [4] that performs a differential sampling to approximate a categorical random variable. Since sigmoid can be viewed as a special two-class case of softmax, we define $p(\cdot)$ using the Gumbel-Sigmoid trick as:

$$p(\boldsymbol{m}^r) = \frac{\exp((\log\pi_0 + g_0)/T)}{\exp((\log\pi_0 + g_0)/T) + \exp((g_1)/T)}, \tag{5}$$

where π_0 represents $\sigma(\boldsymbol{m}^r)$, g_0 and g_1 are sampled from Gumbel distribution. The temperature T is a hyper-parameter to adjust the range of input values, where choosing a larger value could

avoid gradient vanishing during back-propagation. Note that the output of $p(\boldsymbol{m}^r)$ becomes closer to a Bernoulli sample as T is closer to 0. To represent $p(\boldsymbol{m}^r)$ as binary format $\boldsymbol{m}^b$, we use a hard threshold (i.e., 0.5) during forward-propagation of training.

A.3 Re-train the Released Weights

As a byproduct of the proposed LSG method, the preserved model contains unused weights (i.e., attentive-mask as zero) that could be further re-trained for the current task without any impact on previous tasks. Specifically, let $\boldsymbol{w}_{t-2}$ denotes the preserved weights for task $t-2$ including all the weight from task 1 to $t-2$ and $\boldsymbol{w}_{t-2:t-1}$ denote the grown weights for task $t-1$. As we adapt a new kernel-wise attentive-mask $\boldsymbol{m}^a_{t-1}$ for the grown weights, $\boldsymbol{w}_{t-2:t-1} \odot \boldsymbol{m}^a_{t-1}$ is utilized by task $t-1$, while the rest $\boldsymbol{w}_{t-2:t-1} \odot (\mathbb{1} - \boldsymbol{m}^a_t)$ can be released to learn the current task t, where $\mathbb{1}$ is an identity tensor.

In addition, it is worthy to note that these two techniques, selective masking and re-train the released weights, work independently on different parts of the weights, and hence can be jointly optimized in a single training process. The loss function can be formulated as:

$$\mathcal{L}_t(\boldsymbol{w}_t; \boldsymbol{x}_t) = \mathcal{L}_t(f(\{\boldsymbol{w}_{t-2}, \boldsymbol{w}_{t-2:t-1} \odot (\mathbb{1} - \boldsymbol{m}^a_{t-1})\}; \boldsymbol{x}_t), \boldsymbol{y}_t). \tag{6}$$

As shown in Figure 3, after 'Pick and reuse' stage through selective masking and re-training the released weight, if the current accuracy requirement is met, the network does not need grow. Otherwise, it will grow the current model $\boldsymbol{w}_{t-1}$ using the proposed learnable sparse growth method to expand the task-specific features. Specifically, we fix $\boldsymbol{w}_{t-1}$ and learn the grown-mask $\boldsymbol{m}^g_t$ and attentive-mask $\boldsymbol{m}^a_t$ for current task t by following Equation (2). Meanwhile, for the preserved weights, we also adopt the selective masking to re-train the selective-mask and the released weights at the same time.

A.4 Experimental Settings.

We evaluate our CL-LSG on multiple datasets against state-of-the-art continual learning methods. **1) CIFAR-100 Split**. CIFAR-100 [6] consists of images from 100 generic object classes. We split the classes into 10 group, and consider 10-way multi-class classification in each group as a single task. We use training/validation/test splits of 4000/1000/1000 samples. We use a modified version of LeNet-5 with 20-50-800-500 neurons as the base model and train 20 epochs for each task sequentially. **2) CIFAR-100 Superclass**. We divide the CIFAR-100 dataset into 20 tasks. Each task has 5 classes, 2500 training images, and 500 testing images. In the experiment, VGG16-BN model (VGG16 with batch normalization layers) is employed to train the 20 tasks sequentially. **3) MiniImageNet Split**. We divide the MiniImageNet dataset into 20 tasks. Each task has 5 classes, 2375 training images and 500 testing images. We use ResNet18 to train the 20 tasks sequentially.

A.4.1 Comparison with Competing Methods.

To test the efficacy of our method, we compare it with several representative methods using the same task sequence in three categories: **1) Baselines**: We adapt two regular training schemes as follows and select the best accuracy as the target for our method. (i) *scratch*: we train the model from scratch for each task individually using the dataset of current task; (ii) *fine-tune*: we train the model from scratch for the first task only and then fine-tune it for the rest tasks individually. **2) Grow only**: This method only grows the model for each task without affecting the pre-trained backbone part during training. We compare with two representative works, PNN [8] which linearly grows the model for each task, and Grow [18] which grows the model by learnable mask as mentioned earlier. **3) Grow-and-prune**: This method gradually grows and prunes the backbone model for each new task. We choose two representative works for comparison, DEN [17] and CPG [3]. In addition, we also compare with APD [16], which grows the parameter size through involving the task-specific parameters with L1 norm constrain.

A.4.2 Detailed training hyper-parameters

In all experiments, the initial seed network is set to be 1/8 of the full size network, by selecting 1/8 of the channels uniformly for each layer. The experiments are conducted similar training settings as CPG [3]. Specifically, for CIFAR-100, we use the SGD optimizer for weight parameters with 0.01 initial learning rate, and Adam optimizer for the learnable mask with 5e-4 initial learning rate.

We run each task on 50 epochs with batchsize 32. The hyper-parameter λ is set to 1e-4 for all the experiments. We conduct the all the experiments by using a single Nvidia RTX 5000 GPU. For a fair comparison, all the methods are trained on the single NVIDIA Quadro RTX 5000 GPU with the same batch size (i.e, 32).

A.5 Results on CIFAR-100 Split 10 tasks

Table 4: Accuracy on CIFAR-100 Split 10 tasks

Methods	1	2	3	4	5	6	7	8	9	10	**Avg**	Model Size
Scratch	62.4	57.0	67.8	64.4	67.8	67.2	68.6	67.6	61.4	69.6	63.75	10x
PNN	58.2	45.6	59.4	46.8	53.2	58.0	61.2	60.4	53.8	62.4	55.90	1.7x
Grow	56.4	45.4	56.6	47.8	53.2	55.6	61.8	57.4	53.2	59.8	54.72	1.3x
DEN	**61.3**	49.8	60.2	49.4	54.5	57.1	64.3	62.7	55.1	66.5	58.09	1.8x
CPG	59.4	**55.2**	64.8	57.4	59.6	59.4	**66.2**	62.8	57.6	65.4	60.12	1.3x
APD	55.1	52.5	62.3	56.7	**61.4**	58.2	63.4	**63.1**	54.8	64.2	57.54	1.3x
Ours	57.6	53.4	**65.2**	**58.8**	60.4	**60.6**	65.4	62.2	**58.0**	**66.4**	**60.83**	**1.2x**

Table 5: Model size on CIFAR-100 Split 10 tasks

Methods	1	2	3	4	5	6	7	8	9	10
Scratch	1x	2x	3x	4x	5x	6x	7x	8x	9x	10x
PNN	1x	1.08x	1.16x	1.24x	1.32x	1.40x	1.48x	1.54x	1.62x	1.7x
Grow	0.5x	0.64x	0.78x	0.86x	0.92x	0.99x	1.06x	1.16x	1.24	1.3x
DEN	1.0x	1.2x	1.46x	1.46x	1.64x	1.64x	1.8x	1.8x	1.8x	1.8x
CPG	1.0x	1.3x	1.3x	1.3x	1.3x	1.3x	1.3x	1.3x	1.3x	1.3x
APD	1.0x	1.05x	1.10x	1.13x	1.16x	1.19x	1.23x	1.26x	1.28x	1.3x
Ours	**0.5x**	**0.64x**	**0.78x**	**0.78x**	**0.84x**	**0.90x**	**0.90x**	**1.10x**	**1.15x**	**1.2x**

A.6 Results on miniImageNet dataset

Table 6: Accuracy on miniImageNet split 20 tasks

Methods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Avg	Model Size
Scratch	66.0	69.2	65.4	63.2	70.6	73.0	67.0	64.6	65.8	64.4	66.2	68.2	63.2	70.0	66.2	69.6	70.2	66.8	68.6	65.2	67.5	20x
Grow	63.8	61.8	60.0	57.4	63.6	65.2	57.0	56.4	58.4	56.0	57.4	60.6	53.8	65.4	61.8	62.8	58.4	59.4	64.0	55.8	60.1	1.5x
DEN	65.4	65.0	64.6	60.0	64.8	68.6	62.4	59.8	59.4	60.2	59.4	64.6	55.8	63.8	63.4	60.4	65.2	59.6	65.8	59.2	62.5	1.9x
CPG	64.6	**68.6**	64.2	60.4	65.0	69.2	**63.8**	62.6	62.2	**63.4**	66.0	65.2	59.8	**67.0**	63.2	65.8	61.6	64.0	**67.4**	60.4	64.2	1.5x
Ours	**65.8**	68.2	**65.4**	**61.6**	**68.8**	**71.2**	**63.8**	**63.6**	**64.2**	62.8	**66.4**	**66.2**	**60.6**	66.0	**65.2**	**66.8**	**63.2**	**65.6**	65.8	**63.4**	**65.4**	1.5x

Table 7: Model size on miniImageNet split 20 tasks

Methods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Model Size
Scratch	1x	2x	3x	4x	5x	6x	7x	8x	9x	10x	11x	12x	13x	14x	15x	16x	17x	18x	19x	20x	20x
Grow only	0.35x	0.44x	0.52x	0.60x	0.65x	0.65x	0.73x	0.80x	0.88x	0.95x	1.14x	1.14x	1.22x	1.28x	1.28x	1.34x	1.40x	1.40x	1.45x	1.5x	1.5x
DEN	1.0x	1.05x	1.11x	1.16x	1.16x	1.23x	1.39x	1.39x	1.45x	1.45x	1.50x	1.55x	1.59x	1.59x	1.64x	1.64x	1.70x	1.76x	1.82	1.9x	1.9x
CPG	1.0x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x	1.5x
Ours	**0.35x**	**0.44x**	**0.52x**	**0.60x**	**0.65x**	**0.65x**	**0.73x**	**0.80x**	**0.88x**	**0.95x**	**1.14x**	**1.14x**	**1.22x**	**1.28x**	**1.28x**	**1.34x**	**1.40x**	**1.40x**	**1.45x**	**1.5x**	1.5x

A.7 CL-LSG Algorithm

A.8 Ablation Study and Analysis

Table 8: Ablation study on CIFAR-100 Superclass 20 tasks

Methods	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	**Avg.**
Grow only	67.0	73.8	74.4	75.2	81.4	81.2	78.8	80.4	80.6	85.4	85.8	80.4	81.2	80.6	50.8	68.8	66.4	68.2	84.2	88.4	76.5
+Sel-mask	+0.0	+2.0	+3.4	+4.0	+3.4	+4.2	+3.6	+2.0	+3.8	+3.0	+3.8	+3.0	+3.6	+3.4	+2.8	+4.0	+3.2	+3.8	+3.0	+3.6	+3.7
+LSG	+0.2	+0.6	+1.0	+1.8	+1.4	+0.8	+1.2	+0.6	+2.2	+0.6	+1.0	+0.6	+2.0	+0.4	+1.4	+0.6	+0.6	+1.0	+1.4	+1.6	+1.3
+Re-train	+0	+0.4	+0.8	+0.4	+0.4	+0.6	+0.6	+0.8	+1.0	+0.4	+0.8	+0.6	+0.4	+1.2	+0.4	+0.2	+1.0	+0.8	+0.6	+1.0	+0.7
Ours	67.2	76.8	79.6	81.8	86.8	86.8	84.2	83.8	87.8	89.4	91.0	84.6	87.2	85.0	55.4	73.6	71.2	73.8	89.2	94.6	82.2

The Effect of each technique in CL-LSG. We study the effectiveness of each technique in CL-LSG on CIFAR-100 Superclass setting. As shown in Table 8, we consider four different combinations to

Algorithm 1 CL-LSG

Input: Given a pre-defined model with initialized weights $\boldsymbol{w}$, grown-mask $\boldsymbol{m}^g$, kernel-wise attentive-mask $\boldsymbol{m}^a$, selective-mask $\boldsymbol{m}^s$

if *task t == 1* **then**
  Set a target accuracy for task 1
  Apply the attentive-mask $\boldsymbol{m}_1^a$ and grown-mask $\boldsymbol{m}_1^g$ to sparse grow the current model
else
  for *task t = 2 ... T* **do**
    Set a target accuracy for task t
    Apply the selective-mask $\boldsymbol{m}_t^s$ for fixed $\boldsymbol{w}_{t-1}$, and retrain the released weights $\boldsymbol{w}_{t-1}$
    if *Current accuracy < Target accuracy* **then**
      Apply the attentive-mask $\boldsymbol{m}_t^a$ and grown-mask $\boldsymbol{m}_t^g$ to sparse grow the model $\boldsymbol{w}_{t-1}$ to obtain $\boldsymbol{w}_t$
    else
      Set current model as $\boldsymbol{w}_t$
    end
  end
end

perform this ablation study. 1) *Grow only*: only grow the model for each task by learnable mask [18], without any updating on the pre-trained part; 2) *Sel-mask*: adapt the selective masking technique on top of the grow only; 3) *LSG*: replace the grow only with our proposed learnable sparse growth; 4) *Re-train*: further add the re-training of the released sparse weights technique to show the final version. First, it can be seen that the selective-masking method largely improves the accuracy for each task, which shows that simply masking unimportant weights could achieve high adaption ability. Second, our proposed LSG method can not only improve the accuracy, but also reduce the model size by involving kernel-wise sparsity. Last, benefiting from LSG which masks the room to re-train the released sparse weights, it could further improve 0.7% accuracy on average for all tasks.

A.9 Proof of Proposition 1

The conventional learnable growth as shown in Equation (1) is equal to multiply an identity tensor:

$$\begin{aligned}&\mathcal{L}(\boldsymbol{w}, \boldsymbol{m}^g, \boldsymbol{m}^a = \mathbb{1}; \boldsymbol{x})\\ =&\mathcal{L}_t(f(\{\boldsymbol{w} \odot \boldsymbol{m}^g \odot \mathbb{1}\}; \boldsymbol{x}), \boldsymbol{y}) + \lambda||\boldsymbol{m}^g||_0.\end{aligned}$$

For ease of exposition, let $\tilde{\boldsymbol{w}} = (\boldsymbol{w}, \boldsymbol{m}^g)$. Denote

$$\begin{aligned}(\tilde{\boldsymbol{w}}_1^*, (\boldsymbol{m}^a)_1^*) &= \arg\min \mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a);\\ \tilde{\boldsymbol{w}}_2^* &= \arg\min \mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a = \mathbb{1}).\end{aligned}$$

Then, it follows that

$$\begin{aligned}&\mathcal{L}(\tilde{\boldsymbol{w}}_1^*, (\boldsymbol{m}^a)_1^*)\\ \leq &\min\{\mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a|\boldsymbol{m}^a = \mathbb{1}), \mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a|\boldsymbol{m}^a \neq \mathbb{1})\}\end{aligned}$$

for any $\tilde{\boldsymbol{w}}, \boldsymbol{m}^a$.

For $\tilde{\boldsymbol{w}} = \tilde{\boldsymbol{w}}_2^*$,

- if there does not exist any $\boldsymbol{m}^a \neq \mathbb{1}$, such that $\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a|\boldsymbol{m}^a \neq \mathbb{1}) \leq \mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a|\boldsymbol{m}^a = \mathbb{1})$, then

$$\begin{aligned}&\mathcal{L}(\tilde{\boldsymbol{w}}_1^*, (\boldsymbol{m}^a)_1^*)\\ \leq &\min\{\mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a|\boldsymbol{m}^a = \mathbb{1}), \mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a|\boldsymbol{m}^a \neq \mathbb{1})\}\\ \leq &\min\{\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a|\boldsymbol{m}^a = \mathbb{1}), \mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a|\boldsymbol{m}^a \neq \mathbb{1})\}\\ = &\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a = \mathbb{1});\end{aligned}$$

- if there exists some $\boldsymbol{m}_2^a \neq \nVdash$, such that $\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}_2^a) \leq \mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a = \nVdash)$, then

$$\begin{aligned}
&\mathcal{L}(\tilde{\boldsymbol{w}}_1^*, (\boldsymbol{m}^a)_1^*) \\
\leq &\min\{\mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a | \boldsymbol{m}^a = \nVdash), \mathcal{L}(\tilde{\boldsymbol{w}}, \boldsymbol{m}^a | \boldsymbol{m}^a \neq \nVdash)\} \\
\leq &\min\{\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a = \nVdash), \mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}_2^a\} \\
= &\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}_2^a) \\
\leq &\mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a = \nVdash).
\end{aligned}$$

Therefore, we can conclude that $\mathcal{L}(\tilde{\boldsymbol{w}}_1^*, (\boldsymbol{m}^a)_1^*) \leq \mathcal{L}(\tilde{\boldsymbol{w}}_2^*, \boldsymbol{m}^a = \nVdash)$.