Achieving a Better Stability-Plasticity Trade-off via Auxiliary Networks in Continual Learning

Sanghwan Kim
Dept of Computer Science
ETH Zürich
sanghwan.kim@inf.ethz.ch

Lorenzo Noci
Dept of Computer Science
ETH Zürich
lorenzo.noci@inf.ethz.ch

Antonio Orvieto
Dept of Computer Science
ETH Zürich
antonio.orvieto@inf.ethz.ch

Thomas Hofmann
Dept of Computer Science
ETH Zürich
thomas.hofmann@inf.ethz.ch

Abstract

In contrast to the natural capabilities of humans to learn new tasks in a sequential fashion, neural networks are known to suffer from *catastrophic forgetting*, where the model's performances drop dramatically after being optimized for a new task. Since then, the continual learning (CL) community has proposed several solutions to equip the neural network with the ability to learn the current task (*plasticity*) while still achieving high accuracy on the old tasks (*stability*). Despite remarkable improvements, the plasticity-stability trade-off is still far from being solved and its underlying mechanism is poorly understood. In this work, we propose *Auxiliary Network Continual Learning* (ANCL), a new method that regularizes the continually learned model with an additional auxiliary network that is solely optimized on the new task. More concretely, the proposed framework materializes in a regularizer that naturally interpolates between plasticity and stability, surpassing strong baselines on image classification datasets. By analyzing the solutions of several CL methods, we further propose a new technique for hyperparamter search technique which dynamically adjusts the regularization parameter to achieve better stability-plasticity trade-off.

1 Introduction

The main objective of continual learning (CL) is to investigate how neural networks can learn from a sequence of task without progressively forgetting what has already been learned, a problem known in the connectionist approach to cognitive science as catastrophic forgetting [32, 11, 28]. To overcome the issue, various methods have been proposed which can be roughly categorized into regularization-based approaches [18, 2, 6, 40], distillation-based approaches [23, 17, 41, 7], structure-based approaches [1, 38, 25, 26], and replay-based approaches [33, 5, 4, 39]. Based on these approaches, recent works utilize an auxiliary network or module in various ways [36, 41, 25, 24] (Appendix A). Notably, *Active Forgetting with synaptic Expansion-Convergence* (AFEC) [36] adds a regularization term calculated on the auxiliary network that allows the model to actively forget conflicting knowledge and retain the information that can be re-used from previous tasks, aiming at achieving a proper plasticity-stability trade-off. Despite the significant advances, the underlying mechanism of this trade-off has received relatively less attention. Thus, in this work, we formally define the adaptation of the auxiliary network and investigate how it affects the stability-plasticity trade-off.

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

More concretely, our main contributions are as follows:

1. We generalize and formalize the framework of *Auxiliary Network Continual Learning* (ANCL) that can apply the auxiliary network to a variety of continual learning (CL) approaches as a plug-in method (Section 3).
2. We empirically show that ANCL can achieve better stability-plasticity trade-off than the existing CL baselines on CIFAR-100 [20] and Tiny ImageNet [22] (Section 4).
3. Based on ANCL, we suggest a new method called *Adaptive ANCL* (Adap-ANCL) that estimates the regularization hyperparameter in a task-dependent fashion and show that it further boosts the performance (Section 4 and Appendix G).

2 Related Work

We consider the standard problem of learning sequentially T tasks using a neural network f_θ, where $\theta \in \mathbb{R}^P$ denotes the learnable weights. In the standard continual learning (CL) framework, when presented with task t the user has an access to the previous network weights $\theta^*_{1:t-1} \in \mathbb{R}^P$, which are the result of the continual learning from task 1 to $t-1$. It is well known that simply starting optimization from the weights $\theta^*_{1:t-1}$ to obtain the weights $\theta^*_{1:t}$ using the new data from the task t results in catastrophic forgetting [28] of the old tasks. A common way to mitigate catastrophic forgetting is to include a regularization term, which binds the dynamics of each network parameter θ_i ($i \in 1, \dots, P$) to the corresponding old network parameter $\theta_{1:t-1,i}$ through a regularization term $\Omega_{1:t-1,i} > 0$. The new optimization problem then returns:

$$\theta^*_{1:t} = \underset{\theta=(\theta_1,\dots,\theta_P)}{\arg\min} \left[L_{\text{reg}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{2} \sum_i \Omega_{1:t-1,i} (\theta_i - \theta^*_{1:t-1,i})^2 \right], \tag{1}$$

where in classification problems $\mathcal{L}_{\text{CE}}(\theta)$ is cross-entropy loss on the data of the task t and λ is the regularization strength and is usually selected via a grid search procedure. Various regularization-based methods chose different ways to estimate the parameter Ω_i. For example, *Elastic Weight Consolidation* (EWC) [18] calculates Ω_i through the approximation of Fisher Information Matrix (FIM), and *Memory Aware Synapses* (MAS) [2] measures Ω_i as a function of the magnitude of the updates on parameter i. In Appendix B, we further detail these and other CL frameworks that we build upon.

Our approach stems from Wang et al. [36], who uses a biologically inspired argument to propose *Active Forgetting with synaptic Expansion-Convergence* (AFEC). AFEC adds an additional regularization term to the EWC loss to promote *active forgetting*, derived from a Bayesian analysis:

$$\mathcal{L}_{\text{AFEC}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{2} \sum_i \Omega_{1:t-1,i} (\theta_i - \theta^*_{1:t-1,i})^2 + \frac{\lambda_a}{2} \sum_i F_{t,i} (\theta_i - \hat{\theta}_{t,i})^2. \tag{2}$$

The first two terms are the same as above, while the last term promotes active forgetting by regularizing the weights $\theta \in \mathbb{R}^P$ towards the biologically inspired *expanded weights* $\hat{\theta}_t \in \mathbb{R}^P$ (see precise definition in [36]) through the FIM F_t on the task t. The last term in Eqn. (2) can be applied to other regularization-based methods as a *plug-and-play* method.

3 Method

Motivated by AFEC [36], we formally define *Auxiliary Network Continual Learning* (ANCL) which applies the auxiliary network trained on the current task to the continually learned previous network in order to resolve the stability-plasticity dilemma. Fig. 1 illustrates the conceptual difference between CL and ANCL where CL can be any continual learning method that includes a regularizer via the old network. Original CL approaches mainly focus on retaining the old knowledge obtained from the previous tasks by preventing weight updates that take away from the previous weight $\theta^*_{1:t-1}$. However, this might harmfully restrict the model's ability to learn the new knowledge, which will ruin the right balance between stability and plasticity. On the contrary, ANCL maintains the two types of network to maintain this balance: (1) the auxiliary network θ^*_t, which is optimized solely on the current task t allowing for forgetting (*plasticity*) and (2) the old network $\theta^*_{1:t-1}$ that has been sequentially trained until task $t-1$ (*stability*). Then, both networks regularize together the optimization process of the main network weight θ^{ANCL}_{t-1} where the stability-plasticity balance is adjusted through λ and λ_a.

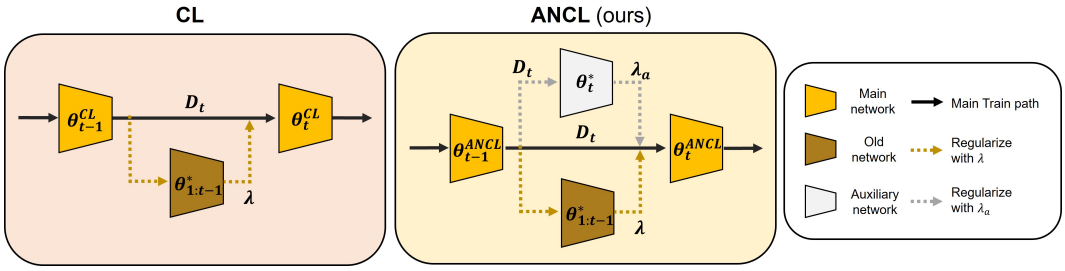

Figure 1: Conceptual comparison of CL and ANCL (ours). 1) CL: the previous weight θ^{CL}_{t-1} is freezed in the old network as $\theta^*_{1:t-1}$ and the old network regularizes the main training via λ. 2) ANCL: the auxiliary network initialized by θ^{ANCL}_{t-1} is trained on the dataset D_t and then freezed as θ^*_t. It regularizes the main training via λ_a in addition to the regularization of the old network.

For example, ANCL can be applied to EWC [18] and MAS [2] to build *Auxiliary Network EWC* (A-EWC) and *Auxiliary Network MAS* (A-MAS) accordingly. Then, the new loss of A-EWC and A-MAS can be written as follows:

$$\mathcal{L}_{\text{A-reg}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{2}\sum_i \Omega_{1:t-1,i}(\theta_i - \theta^*_{1:t-1,i})^2 + \frac{\lambda_a}{2}\sum_i \Omega_{t,i}(\theta_i - \theta^*_{t,i})^2, \tag{3}$$

where the importance Ω_t of the auxiliary parameter $\theta^*_{t,i}$ is calculated following the original methods (EWC or MAS). The first two terms are equal to Eqn. (1) and A-EWC is equivalent to AFEC[1] except that the importance Ω_t of the auxiliary network in ANCL is calculated only once[2] before the training of the new task and then fixed afterward.

Similarly, ANCL can be extended to distillation-based methods such as *Learning without Forgetting* (LwF) [23] or *less-forgetting learning* (LFL) [17]. Given the data x_j, LwF regularizes the logit y_j of the current model to be similar to the logit $y_{old,j}$ of the old network, while LFL drifts the activation $f(x_j)$ before the last layer of the main network towards the activation $f_{old}(x_j)$ of the old network (detailed in Appendix B). By applying ANCL to LwF and LFL, the new loss of *Auxiliary Network LwF* (A-LwF) and *Auxiliary Network LFL* (A-LFL) is written as follows:

$$\mathcal{L}_{\text{A-LwF}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda\sum_j\sum_{c=0}^{C_{1:t}} -y^c_{old,j}\log y^c_j + \lambda_a\sum_j\sum_{c=0}^{C_{1:t}} -y^c_{aux,j}\log y^c_j, \tag{4}$$

$$\mathcal{L}_{\text{A-LFL}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda\sum_j ||f(x_j) - f_{old}(x_j)||^2_2 + \lambda_a\sum_j ||f(x_j) - f_{aux}(x_j)||^2_2. \tag{5}$$

In Eqn. (4), y_{aux} represents the temperature-scaled logit of the auxiliary network and $C_{1:t}$ denotes the total number of classes until task t. The new regularizer is double summated over the class position c and the data x_j. In Eqn. (5), $f_{aux}(x)$ is the normalized and centered activation of the auxiliary network.

The auxiliary network θ^*_t of ANCL works similarly to the expanded parameter $\hat{\theta}_t$ of AFEC with respect to adding an extra loss term, but ANCL uses the *method-dependent* regularizer compared to the *fixed and independent* regularizer of AFEC based on FIM. In other words, while AFEC plugs in the same loss term of the expanded parameter to every method, ANCL generates the loss term of the auxiliary network in the same way as the original CL where ANCL is applied. This is a more natural way to implicitly embrace two networks because the stability-plasticity trade-off can be easily controlled by the scaling hyperparameters (λ and λ_a in Eqn. (3)) of two regularizers in the same type. This is mathematically shown in Appendix C, where we analyze the gradients of ANCL. If the two regularizers are in different type, each regularizer will change in different magnitude and consequently make it hard to arrive at the equilibrium.

Table 1: Averaged accuracy (%) on (1) CIFAR-100/10 (upper row) and (2) Tinyimagenet-200/10 (lower row), averaged by 3 different seeds with error bar ($\pm$ standard error).

Methods	Fine-tuning (Lowerbound)	Joint (Upperbound)	EWC [18]	MAS [2]	LwF [23]	LFL [17]
CL (original)	$38.90_{\pm 1.59}$	$89.64_{\pm 0.37}$	$58.13_{\pm 0.87}$	$60.56_{\pm 0.82}$	$78.87_{\pm 0.69}$	$74.50_{\pm 0.57}$
ANCL (ours)	n/a	n/a	$60.86_{\pm 1.46}$	$64.43_{\pm 1.17}$	$79.42_{\pm 0.57}$	$75.23_{\pm 0.67}$
Adap-ANCL (ours)	n/a	n/a	$61.53_{\pm 1.16}$	$64.87_{\pm 1.32}$	$80.16_{\pm 0.91}$	$76.60_{\pm 0.57}$
CL (original)	$28.51_{\pm 0.75}$	$67.98_{\pm 1.15}$	$50.10_{\pm 0.78}$	$49.50_{\pm 1.18}$	$59.04_{\pm 0.62}$	$60.20_{\pm 0.66}$
ANCL (ours)	n/a	n/a	$52.49_{\pm 0.71}$	$50.11_{\pm 1.09}$	$60.96_{\pm 0.76}$	$61.32_{\pm 0.68}$
Adap-ANCL (ours)	n/a	n/a	$52.85_{\pm 0.68}$	$50.91_{\pm 0.98}$	$61.53_{\pm 0.72}$	$62.50_{\pm 0.57}$

4 Experiment

CIFAR-100 [20] and Tiny ImageNet [22] are chosen to form two benchmarks after divided into 10 tasks each: (1) CIFAR-100/10 and (2) Tinyimagenet-200/10. Benchmark (1) and (2) contain 10 and 20 classes per task respectively. Resnet32 [14] with multi-head at the last layer is used for all experiments. Our experiment is classified as task incremental learning [34], where the task identity is given during both the training session and inference. In addition, every experiment is conducted 3 times, and we report the averaged metric.

We evaluate our ANCL approaches in Table 1: A-EWC, A-MAS, A-LwF, and A-LFL. Fine-tuning is the naive approach that the model is fine-tuned on a sequence of task which is regarded as a lowerbound. Joint uses the whole datasets to train the model, which then becomes an upperbound. In all methods, applying ANCL gives an extra boost in accuracy compared to standard CL. This coincides with the theoretical analysis of the ANCL loss in Appendix C, supporting that ANCL is more capable of balancing stability and plasticity. The grid search is performed on both λ and λ_a (detailed in Appendix E), and the accuracy of all tasks can be seen in Appendix D.

Moreover, we observed that the model favors more stability over plasticity during the training of later tasks through extensive analyses of the stability-plasticity trade-off (Appendix F). One possible reason is that the model should remember more previous knowledge as the training progresses. Based on this finding, we propose to employ the task-dependent hyperparameter $\lambda_{a,t}$ instead of λ_a called *Adaptive ANCL* (Adap-ANCL) which is defined as follows:

$$\lambda_{a,t} = \frac{C_t}{C_{1:t}} \lambda_{a,init} \tag{6}$$

C_t denotes the number of classes at task t and $C_{1:t}$ represents all classes accumulated until task t. $\lambda_{a,init}$ is an initial value determined by grid search. Thus, the adaptive parameter $\lambda_{a,t}$ decreases as the model learns more task. In such manner, ANCL is dynamically optimized toward the better balance between stability and plasticity in every task. Adap-ANCL further improves the accuracy of ANCL, which can be seen in Table 1.

5 Conclusion

In this paper, we propose a novel CL scheme called ANCL which effectively employ the auxiliary network to reach better stability-plasticity trade-off as a plug-in method. It is shown theoretically and empirically that ANCL has a high potential and is actually able to effectively merge the old and auxiliary networks. Based on ANCL, we finally propose Adap-ANCL which actively modifies a key hyperparameter depending on task.

[1]However, A-MAS is not equal to AFEC as the new regularizer of ANCL is based on MAS importance.

[2]The code implementation of AFEC newly calculates F_t every epoch.

References

[1] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020.

[2] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[3] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[4] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[5] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248, 2018.

[6] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[7] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019.

[8] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020.

[9] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[10] J. Frankle, G. K. Dziugaite, D. Roy, and M. Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[11] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[12] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[13] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[16] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019.

[17] H. Jung, J. Ju, M. Jung, and J. Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[19] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[20] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] R. Kuditipudi, X. Wang, H. Lee, Y. Zhang, Z. Li, W. Hu, R. Ge, and S. Arora. Explaining landscape connectivity of low-cost solutions for multilayer nets. *Advances in neural information processing systems*, 32, 2019.

[22] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

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

[24] G. Lin, H. Chu, and H. Lai. Towards better plasticity-stability trade-off in incremental learning: A simple linear connector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2022.

[25] Y. Liu, B. Schiele, and Q. Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2544–2553, 2021.

[26] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[27] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *arXiv preprint arXiv:2010.15277*, 2020.

[28] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[29] S. I. Mirzadeh, M. Farajtabar, D. Gorur, R. Pascanu, and H. Ghasemzadeh. Linear mode connectivity in multitask and continual learning. *arXiv preprint arXiv:2010.04495*, 2020.

[30] S. I. Mirzadeh, M. Farajtabar, R. Pascanu, and H. Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33: 7308–7320, 2020.

[31] V. V. Ramasesh, E. Dyer, and M. Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. *arXiv preprint arXiv:2007.07400*, 2020.

[32] R. Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[33] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[34] G. M. Van de Ven and A. S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[35] L. Venturi, A. S. Bandeira, and J. Bruna. Spurious valleys in one-hidden-layer neural network optimization landscapes. *Journal of Machine Learning Research*, 20:133, 2019.

[36] L. Wang, M. Zhang, Z. Jia, Q. Li, C. Bao, K. Ma, J. Zhu, and Y. Zhong. Afec: Active forgetting of negative transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22379–22391, 2021.

[37] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.

[38] S. Yan, J. Xie, and X. 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.

[39] J. Yoon, D. Madaan, E. Yang, and S. J. Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021.

[40] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[41] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020.

[42] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S.-T. Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217, 2020.

A Continual Learning Methods using Auxiliary Network

In the last few years, several papers have proposed to use an auxiliary network or an extra module which is trained on a new task. They tried to combine it with a previous network or module which has been trained on old tasks using existing continual learning techniques.

Elastic Weight Consolidation (EWC) [18] is one of the initial works among weight regularization methods that regularize weights related to the previous tasks. After training each task, EWC copies and freezes the current network as an old network. Then, it estimates the importance of each parameter in the old network so that the weights with the high importance remain unchanged when the model is optimized on future task. Based on the EWC, Wang et al. [36] proposes *Active Forgetting with synaptic Expansion-Convergence* (AFEC) which further regularizes the weights relevant to the current task through expanded parameters. The expanded parameters are solely trained on the dataset of the new task initialized by the old network and are allowed to forget the previous tasks. As a result, AFEC can reduce potential negative transfer in continual learning by selectively merging the old parameters with the expanded parameter. The stability-plasticity balance in AFEC is adjusted via hyperparameters which scale the regularization terms for remembering the old tasks and learning the new tasks, respectively.

Learning without Forgetting (LwF) [23] prevents forgetting using knowledge distillation [3, 15]. They apply a distillation loss so that the model can learn soft targets generated by the old model instead of typical one-hot targets. The old model is copied and freezed before the training of the current task like EWC. *Deep Model Consolidation* (DMC) [41] is another distillation method built upon LwF. DMC proposes double distillation loss where the soft targets are generated using both the old model and a new model. The new model in DMC is basically the same concept as the expanded parameter in AFEC which is optimized on the current task. Then, the logits of new classes from the new model and the logits of old classes from the old model are concatenated to build the final soft targets. Then, the stability-plasticity balance is achieved by penalizing the model to generate the same output as the old model for old classes and the same output as the new model for new classes.

Adaptive Aggregation Networks (AANet) [25] explicitly expands ResNet [14] to have the two types of residual blocks at each residual level: the one for retaining old knowledge and the other for learning new knowledge. The outputs from the two residual blocks are linearly combined by aggregation weights and then proceeded to the next-level layer. They train AANets through bilevel optimization. In the first level, the parameters of the two residual blocks are trained. In the second level, the aggregation weights are adapted which decide the balance between stability and plasticity within the ResNet.

Recent work by Mirzadeh et al. [29] observes that multitask and continual solutions are connected by very simple curves with a low error in weight space, which is called *Linear Mode Connectivity*. They empirically prove that this connectivity is a linear path if the multitask learning and the continual learning share same initialization weight. Based on this observation, Lin et al. [24] proposes a simple linear connector that linearly add the weights of two networks following this linear path to emulate the multitask solution: the one model remembering the old tasks and the other model learning the new tasks. The stability-plasticity balance is preserved by combining the two networks.

The above methods [36, 41, 25, 24] all share the property that the auxiliary model or module is used to solve the stability-plasticity dilemma in continual learning. Consequently, these methods are able to learn the current task better than the original method while still retaining the knowledge of the previous tasks. However, the underlying mechanism of the interaction between the previous model and the auxiliary model is not widely studied. Therefore, in this paper, we first formalize the framework of continual learning that adopts the auxiliary network called *Auxiliary Network Continual Learning* (ANCL). Given this environment, we investigate the stability-plasticity trade-off from both a theoretical and empirical point of view and perform various analyses to better understand it.

B Detail Explanation of Existing Continual Learning approaches

B.1 Weight Regularization Methods

Regularization-based methods in continual learning regularize the weight of the main network toward the old network so that the current weight does not deviate a lot from the previous optimal weight. Various approaches choose different ways to calculate the regularizing parameter Ω_i in Eqn. (1).

For example, *Elastic Weight Consolidation* (EWC) [18] calculates Ω_i through the approximation of Fisher Information Matrix (FIM). The diagonal elements of FIM quantifies how curved the likelihood of each parameter is and can be approximated by a Hessian matrix near the optimum. In other words, the larger the change in the gradient, the more relevant the corresponding parameter is to the previous tasks. FIM is calculated after training each task, which means that R_i of EWC cannot fully reflect the learning trajectory of each network weight.

Compared to EWC, *Memory Aware Synapses* (MAS) [2] proposes accumulating the changes of each parameter throughout the update history. R_i is measured through the magnitude of the updates on each parameter according to the change in output. In other words, if the small change of specific parameter in a network causes the huge change of an output, that parameter should be memorized first.

Based on the above regularization-based methods, Wang et al. [36] suggests a biologically inspired argument to propose *Active Forgetting with synaptic Expansion-Convergence* (AFEC) where an additional regularization term is added on EWC loss. They argue that this term stimulates the active forgetting of previous knowledge that interferes with the new knowledge, whereas the existing regularization-based approaches highly concentrate on retaining the old weight. The new loss of AFEC can be seen in Eqn. (2). The last term in Eqn. (2) promotes active forgetting by regularizing the parameters $\theta \in \mathbb{R}^P$ towards the biologically inspired *expanded parameters* $\hat{\theta}_t \in \mathbb{R}^P$ (see precise definition in [36]) through FIM F_t on task t. The expanded parameter is solely trained on the current dataset allowing the forgetting of the old datasets and λ_a is a hyperparameter determined by grid search. As a result, the main network parameter θ efficiently learns from both the old parameters $\theta^*_{1:t-1}$ and the expanded parameters $\hat{\theta}_t$. Moreover, the last term of Eqn. (2) can be applied to other regularization-based methods as a *plug-and-play* method.

The expanded parameters $\hat{\theta}_t$ in AFEC are computed as follows: at the beginning of each task, the expanded parameter is initialized by the old parameters, Then, at each epoch, it is trained on task t without regularization, returning the network parameters $\hat{\theta}_t$. For the current epoch of the original network, these weights are then taken into account to modify the regularized dynamics. This procedure is repeated every epoch, and hence for each epoch the dynamics of SGD is augmented with a freshly computed regularizer.

B.2 Knowledge Distillation Methods

Distillation-based approaches prevent forgetting through knowledge distillation [3, 15] which was originally devised to train a smaller student network from a larger teacher network. In this way, the main network can emulate the activation or logit of the previous network while learning the current task.

Learning without Forgetting (LwF) [23] propose following loss to make current network to distill the knowledge of the old network via imitating the logit:

$$\mathcal{L}_{\text{LwF}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda \sum_{j} \sum_{c=0}^{C_{1:t-1}} -y^c_{old,j} \log y^c_j. \tag{7}$$

Similar to Eqn. (1), θ is the weight of the main network that is trained on a sequence of tasks so far and $\mathcal{L}_{\text{CE}}(\theta)$ is cross-entropy loss on current task t. y_{old} and y are temperature-scaled logits of the old network and the current network, respectively. The main network that continuously learns data from task 1 to task $t-1$ is freezed and save as the old network. $C_{1:t-1}$ denotes the total number of classes until task $t-1$. Thus, y^c_j refers to the cth output of the logit of input x_j. The logit scaled by temperature τ on the cth class position is calculated as follows:

$$y^c_j = \frac{(\boldsymbol{o}(x_j)^c)^{1/\tau}}{\sum_k (\boldsymbol{o}(x_j)^k)^{1/\tau}}, \qquad y^c_{old,j} = \frac{(\boldsymbol{o}(x_j)^c_{old})^{1/\tau}}{\sum_k (\boldsymbol{o}(x_j)^k_{old})^{1/\tau}} \tag{8}$$

where $\boldsymbol{o}(x_j)$ and $\boldsymbol{o}_{old}(x_j)$ are the output of the current network and the old network before softmax is applied.

Another distillation method, *less-forgetting learning* (LFL) [17], penalizes the differences of activations before last layer instead of logits:

$$\mathcal{L}_{\text{LFL}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda \sum_j ||f(x_j) - f_{old}(x_j)||_2^2 \tag{9}$$

$f(x_j)$ and $f_{old}(x_j)$ are centered and normalized activation from the main and old network each. The knowledge of the old network is transfered via the difference of two activations which lead the activation of the current model to the counterpart of the previous model.

C The Role of Auxiliary Network

In this section, we look into our CL and ANCL loss closely by analyzing its gradient. According to each gradient, we compare how CL and ANCL methods converge to optimal weight in the point of stability-plasticity balance.

First, we start with EWC loss [18]:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{2} \sum_i F_{1:t-1,i}(\theta_i - \theta^*_{1:t-1,i})^2 \tag{10}$$

where $F_{1:t-1,i}$ represents the accumulated fisher information matrix of parameter $\theta^*_{1:t-1,i}$ until task $t-1$. For simplicity, let's assume importance F is fixed among all parameters of network. Then, we can simplify the EWC loss as follows:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{2} F_{1:t-1} ||\theta - \theta^*_{1:t-1}||_2^2 \tag{11}$$

where we take the derivative with respect to the weight of the model θ. Then, we have

$$\nabla_\theta \mathcal{L}_{\text{EWC}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \lambda F_{1:t-1}(\theta - \theta^*_{1:t-1}). \tag{12}$$

The gradient consists of two terms: the first term that updates θ towards the minima of the current task through cross-entropy loss and the second term which regularizes θ to be as close as possible to $\theta^*_{1:t-1}$. If λ is gradually increased, the second term becomes more and more dominant in the gradient so that θ converges near $\theta^*_{1:t-1}$ at the end of the training. In this case, stability is achieved but plasticity is neglected.

Next, we analyze with A-EWC loss in the same way. We also assume here that importance F is fixed among all parameters of network. Then, Eqn. (3) can be rewritten as

$$\mathcal{L}_{\text{A-EWC}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{2} F_{1:t-1} ||\theta - \theta^*_{1:t-1}||_2^2 + \frac{\lambda_a}{2} F_t ||\theta - \theta^*_t||_2^2. \tag{13}$$

Take the derivative with respect to θ:

$$\nabla_\theta \mathcal{L}_{\text{A-EWC}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \lambda F_{1:t-1}(\theta - \theta^*_{1:t-1}) + \lambda_a F_t(\theta - \theta^*_t) \tag{14}$$

$$= \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + (\alpha + \beta)(\theta - \frac{\alpha\theta^*_{1:t-1} + \beta\theta^*_t}{\alpha + \beta}). \tag{15}$$

In the last equality, $\alpha = \lambda F_{1:t-1}$ and $\beta = \lambda_a F_t$ are used. The gradient of A-EWC also consists of two parts. The first term is the same as before and the second term penalizes θ to move toward the interpolation of $\theta^*_{1:t-1}$ and θ^*_t. This linear interpolation fully depends on the hyperparameter λ and λ_a as we fixed the importance. Therefore, θ will converge to the weight between $\theta^*_{1:t-1}$ and θ^*_t at the end, which turns out to be the region with high accuracy by mode connectivity figures in Appendix F.4. In Eqn. (13), the balance between stability and plasticity can be adjusted more easily through the ratio between λ and λ_a while it is solely dependent on λ in Eqn. (11).

For MAS [2] and A-MAS, similar analaysis can be applied as above if we simply replace the importance of EWC F with the importance of MAS Ω.

Among distillation losses, we first looked into LFL [17] loss:

$$\mathcal{L}_{\text{LFL}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda \sum_j ||f(x_j) - f_{old}(x_j)||_2^2. \tag{16}$$

Again, take the derivative with respect to θ:

$$\nabla_\theta \mathcal{L}_{\text{LFL}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + 2\lambda \sum_j (f(x_j) - f_{old}(x_j)) \nabla_\theta f(x_j). \tag{17}$$

The gradient of LFL loss can be divided into two parts: the first part that drives θ toward the minima of the current task and the second part that regularizes the difference of activations between the current model and the old model. Compared to Eqn. (12), LFL has more flexibility to memorize previous knowledge as it does not directly regularize its weight but only its activations. The model trained with too high λ will mainly focus on generating exactly same activations as old model while neglecting learning current task.

Next, we investigate A-LFL loss in Eqn. (5):

$$\mathcal{L}_{\text{A-LFL}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda \sum_j ||f(x_j) - f_{old}(x_j)||_2^2 + \lambda_a \sum_j ||f(x_j) - f_{aux}(x_j)||_2^2 \tag{18}$$

Same derivative is applied as follows:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{A-LFL}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) &+ 2\lambda \sum_j (f(x_j) - f_{old}(x_j)) \nabla_\theta f(x_j) \\ &+ 2\lambda_a \sum_j (f(x_j) - f_{aux}(x_j)) \nabla_\theta f(x_j)\end{aligned} \tag{19}$$

Which can be organized as

$$\nabla_\theta \mathcal{L}_{\text{A-LFL}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + 2(\lambda + \lambda_a) \sum_j (f(x_j) - \frac{\lambda f_{old}(x_j) + \lambda_a f_{aux}(x_j)}{\lambda + \lambda_a}) \nabla_\theta f(x_j) \tag{20}$$

The second term of the gradient of A-LFL derives the activation of the main network to be equal to the interpolated activation of the auxiliary and old network. The interpolation is decided by the ratio of λ and λ_a. It is shown in Fig. 11 that this leads the weight to achieve the better trade-off in accuracy landscape overcoming simple linear interpolation between the old and auxiliary weight Moreover, it is worth noting that Eqn. (20) become quite similar form as Eqn. (15) if we assume a one-layer network where $f_{old}(x_j) = W_{old} x_j$, $f_{aux}(x_j) = W_{aux} x_j$, and $f(x_j) = W x_j$ hold such that $W_{old}, W_{aux}, W \in \mathbb{R}^{p \times d}$ and $x_j \in \mathbb{R}^d$:

$$\nabla_\theta \mathcal{L}_{\text{A-LFL}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + 2(\lambda + \lambda_a) \sum_j (W - \frac{\lambda W_{old} + \lambda_a W_{aux}}{\lambda + \lambda_a}) x_j x_j^T. \tag{21}$$

Now, the second term drift model weight W (except the last layer) toward the interpolation of the old weight and the auxiliary weight like that of the A-EWC gradient. Again, we strike a balance between stability and plasticity balance through λ and λ_a.

Next distillation loss is LwF [23] as follows:

$$\mathcal{L}_{\text{LwF}}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \lambda \sum_j \sum_{c=0}^{C_{1:t}} -y_{old,j}^c \log y_j^c \tag{22}$$

For simplicity, we assume that y_{old} and y are softmax logits scaled by temperature τ such as

$$y_j^c = \frac{e^{\boldsymbol{o}(x_j)^c/\tau}}{\sum_k e^{\boldsymbol{o}(x_j)^k/\tau}}, \qquad y_{old,j}^c = \frac{e^{\boldsymbol{o}(x_j)_{old}^c/\tau}}{\sum_k e^{\boldsymbol{o}(x_j)_{old}^k/\tau}}. \tag{23}$$

Then, referring to Hinton et al. [15], the gradient of LwF with respect to logits θ can be calculated as follows:

$$\nabla_\theta \mathcal{L}_{\text{LwF}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{\tau} \sum_j (y_j - y_{old,j}) \nabla_\theta \boldsymbol{o}(x_j). \tag{24}$$

When τ is sufficiently large, then softmax can be approximated using $exp(\boldsymbol{o}(x_j)/\tau) \approx 1 + \boldsymbol{o}(x_j)/\tau$:

$$\nabla_\theta \mathcal{L}_{\text{LwF}}(\theta) \approx \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{\tau} \sum_j \left(\frac{1 + \boldsymbol{o}(x_j)/\tau}{C_{1:t-1} + \sum_k \boldsymbol{o}(x_j)^k/\tau} - \frac{1 + \boldsymbol{o}(x_j)_{old}/\tau}{C_{1:t-1} + \sum_k \boldsymbol{o}(x_j)^k_{old}/\tau} \right) \nabla_\theta \boldsymbol{o}(x_j) \quad (25)$$

We further assume that logits of main network and old network are zero mean, i.e., $\sum_k \boldsymbol{o}(x_j)^k = 0$ and $\sum_k \boldsymbol{o}(x_j)^k_{old} = 0$. Finally, we get the following approximation:

$$\nabla_\theta \mathcal{L}_{\text{LwF}}(\theta) \approx \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{C_{1:t-1}\tau^2} \sum_j (\boldsymbol{o}(x_j) - \boldsymbol{o}(x_j)_{old}) \nabla_\theta \boldsymbol{o}(x_j) \quad (26)$$

This demonstrates that the gradient of LwF is in equivalent form as the gradient of LFL under sufficiently large temperature and zero-mean logits from both main and old network. One should also note that $\boldsymbol{o}(x_j)$ in LwF is logit and $f(x_j)$ in LFL is activation before last layer.

Similarly, we can approximate the gradient of A-LwF:

$$\nabla_\theta \mathcal{L}_{\text{A-LwF}}(\theta) \approx \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda}{C_{1:t-1}\tau^2} \sum_j (\boldsymbol{o}(x_j) - \boldsymbol{o}(x_j)_{old}) \nabla_\theta \boldsymbol{o}(x_j) + \frac{\lambda_a}{C_{1:t-1}\tau^2} \sum_j (\boldsymbol{o}(x_j) - \boldsymbol{o}(x_j)_{aux}) \nabla_\theta \boldsymbol{o}(x_j) \quad (27)$$

Then, we get the final form as follows:

$$\nabla_\theta \mathcal{L}_{\text{A-LwF}}(\theta) \approx \nabla_\theta \mathcal{L}_{\text{CE}}(\theta) + \frac{\lambda + \lambda_a}{C_{1:t-1}\tau^2} \sum_j \left(\boldsymbol{o}(x_j) - \frac{\lambda \boldsymbol{o}(x_j)_{old} + \lambda_a \boldsymbol{o}(x_j)_{aux}}{\lambda + \lambda_a} \right) \nabla_\theta \boldsymbol{o}(x_j) \quad (28)$$

Similar to A-LFL, the second part of gradient shift the logit from main network toward the interpolation of the logits from old and auxiliary network. The stability-plasticity dilemma is solved by directly controlling λ and λ_a.

D Detailed Results on Experiments

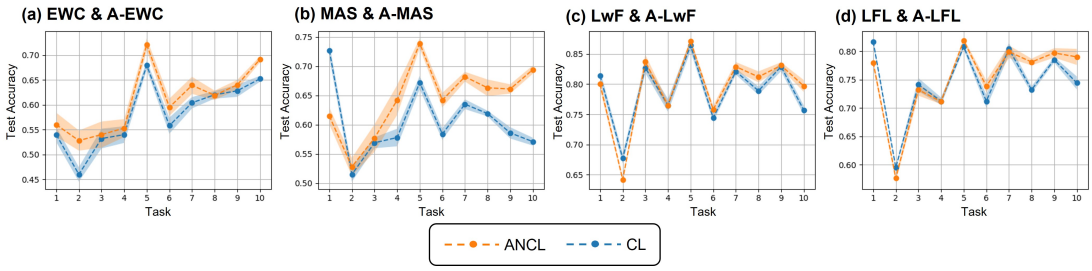

Figure 2: The final accuracy on all tasks of (1) CIFAR-100/10 with its standard error as an error band. The orange line visualizes ANCL methods ((a) A-EWC, (b) A-MAS, (c) A-LwF, and (d) A-LFL) and the blue line represents CL methods ((a) EWC, (b) MAS, (c) LwF, and (d) LFL).

In this section, we plot the final accuracy of the sequential 10 task of CIFAR-100 in Figure 2. In Figure 2, the accuracy for most tasks of ANCL approaches is higher than that of CL approaches. In particular, ANCL achieves better performance in the later task compared to CL at the cost of losing some of the earlier task accuracy, which is well shown in Figure 2 (b) and (c). This is because ANCL methods are able to learn new tasks better than CL through the extra regularization term as explained in Section C. However, ANCL loses a bit of ability to remember initial knowledge as a trade-off, but ANCL is still comparable to CL in earlier tasks.

E Training details

Benchmark: CIFAR-100 [20] and Tiny ImageNet [22] are used to evalulate our ANCL methods. CIFAR-100 contains 60,000 colored images from 100 classes with the size of 32×32 and is divided into 50,000 and 10,000 images to form a train set and a test set. Then, the 10% of the train set is randomly sampled to build a validation set. CIFAR-100 is divided into 10 tasks of 10 classes each to construct a benchmark **(1) CIFAR-100/10**.

Tiny ImageNet consists of 100,000 train images and 10,000 test images and the 10% of the train set is again used as a validation set. It includes 200 classes and all data are 64×64 colored images which are resized as 32×32 for both training and inference. We equally divide Tiny ImageNet into 10 to build a benchmark **(2) TinyImagenet-200/10**.

Architecture: Resnet32 [14] are selected to conduct all experiments which are commonly used by serveral works in the literature of continual learning [33], [37], [16], [42], [8]. Multi-head layer is deployed at the last layer of Resnet32 to generate output with task identity. Thus, total number of head is equal to the number of tasks. Note that the heads of previous tasks are freezed while the model learns the current task, which helps to prevent forgetting compared to baselines updating every parameter [27].

Implementation: We build our ANCL methods based on code implementation of *Framework for Analysis of Class-Incremental Learning* (FACIL) [27] which supports several baselines and existing CL methods. SGD Optimizer with 0.9 momentum and the mini-batch size of 128 is applied to all experiments. The initial learning rate of each task is determined by grid search and the learning rate of the first task is slightly larger than that of the following task as our model is trained from scratch. Then, the learning rate is decreased by factor 3 whenever there is no improvement in validation loss for 5 epochs. If the learning rate reaches minimum threshold, the model stops training. The model can be trained for maximum 200 epochs per task.

Baseline: First, we do not apply any memory buffer that has an access to the previous datasets. Task identity is given during both training and inference, which is regarded as a task incremental learning scenario. In addition, every experiment train the model from scratch and conducted 3 times to generate averaged accuracy.

Finetuning is the naive approach that model is finetuned on each task, which is regarded as a lowerbound. Joint use the whole dataset to train model, which becomes an upperbound. For weight regularization methods, we evaluate EWC [18] and MAS [2]. For distillation methods, we implement LwF [23], and LFL [17]. Based on these CL methods, we also compare our ANCL approaches: A-EWC, A-MAS, A-LwF, and A-LFL.

Evaluation Metric: In our paper, averaged accuracy (AAC) for T task is reported after the training of all tasks:

$$AAC = \frac{1}{T}\sum_{i=1}^{T} A_{T,i} \tag{29}$$

where $A_{j,k}$ is the test accuracy of task k after the continual learning of task j.

Resources: We conduct all our experiments using gpu "NVIDIA GeForce GTX 1080 Ti".

Algorithm: Detailed pseudocode in Algorithm 1 explains how we implement ANCL approaches. This is valid for all ANCL methods (A-EWC, A-MAS, A-LwF, and A-LFL) if appropriate ANCL loss is applied. If lines 9-12 are skipped and "ANCL loss" in line 14 is replaced with "CL loss", Algorithm 1 becomes the original CL algorithm.

F Trade-off Analysis

In this section, three analyses were performed to deeply understand how stability-plasticity dilemma is solved through ANCL. It is worth to note that the specific learning regime described in Appendix

Algorithm 1: ANCL Algorithm

Input: Main network weight θ, Auxiliary network weight θ_t, Old network weight $\theta_{1:t-1}$, Hyperparameters λ, λ_a

Output: Optimal weight θ^*

for *task t = 1, 2, .., N* **do**
  if $t = 1$ **then**
    `// Train main network on first task`
    for *epoch e = 1, 2, .., E* **do**
      Train θ with $\mathcal{L}_{\text{CE}}(\theta)$ to obtain θ^* on task 1
    Set $\theta^*_{1:1} = \theta^*$
    (Calculate the importance of $\theta^*_{1:1}$ if needed)
    `// Save old network weight`
    Freeze and save $\theta^*_{1:1}$
  else
    `// Train auxiliary network`
    θ_t = copy($\theta^*_{1:t-1}$)
    Train θ_t with $\mathcal{L}_{\text{CE}}(\theta_t)$ to obtain θ^*_t
    (Calculate the importance of θ^*_t if needed)
    `// Save auxiliary network weight`
    Freeze and save θ^*_t
    `// Train main network`
    for *epoch e = 1, 2, .., E* **do**
      Train θ with ANCL loss Eqn. (3, 4, 5) to obtain θ^* on task t
    Set $\theta^*_{1:t-1} = \theta^*$
    (Calculate the importance of $\theta^*_{1:t-1}$ if needed)
    `// Save main network weight as old network weight`
    Freeze and save $\theta^*_{1:t-1}$

F.1 Fig. 3 was adopted in all following analyses which is built upon the setting in Mirzadeh et al. [29].

F.1 Training Regime for Trade-off Analysis

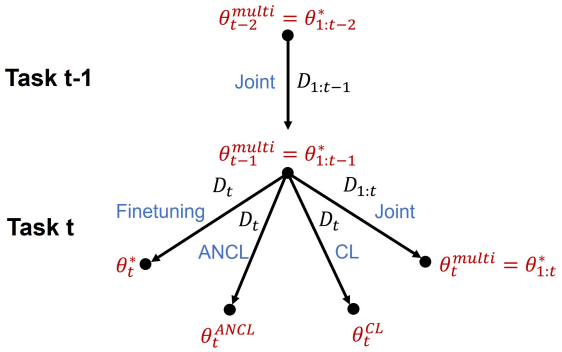

Figure 3: Training regime for analysis. D_t stands for the dataset of task t and $D_{1:t-1}$ means combined dataset from task 1 to task $t-1$.

All analyses in this paper are based on the specific learning scheme described in Fig. 3. On task t, every model starts training from multitask weight θ^{multi}_{t-1} which is trained on combined dataset $D_{1:t-1}$ before (i.e., train on $D_1 \cup D_2 \cup \cdots \cup D_{t-1}$). If we fine-tune the model on data D_t, it becomes auxiliary network θ^*_t which regularizes ANCL later. ANCL and CL approach is applied to get corresponding weitht θ^{ANCL}_t and θ^{CL}_t each. Here, the weight of multiple tasks θ^{multi}_{t-1} on task $t-1$ works as the old

network weight $\theta^*_{1:t-1}$ to regularize ANCL and CL. Finally, the initial weight is trained on the data $D_{1:t}$ to train the next multitask model θ_t^{multi} and it becomes the next initialization(or old network) for the task $t+1$. We used a fixed multitask weight as an initialization at the start of each task for a fair comparison among all methods. Otherwise, every method will have different old and auxiliary network which end up confusing following precise analyses. Especially, this training scheme is essential to visualize mode connectivity figure as Mirzadeh et al. [29] empirically shows that linear mode connectivity is valid when models share same initialization.

F.2 Weight Distance

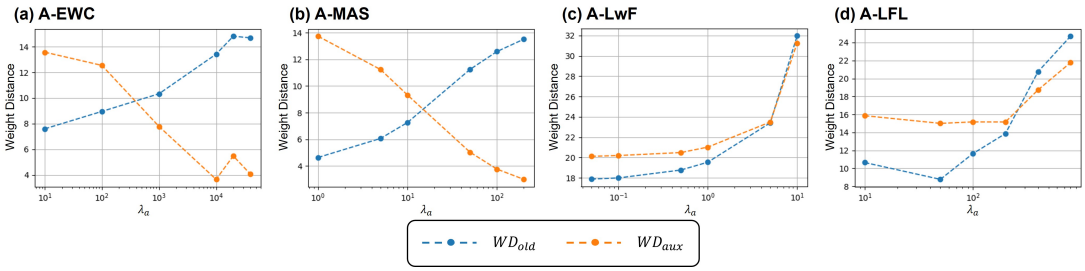

Figure 4: Weight distance for four ANCL methods on CIFAR-100 when $t = 2$ in Eqn. (32). We plot the weight distance on y-axis according to six different λ_a on x-axis while λ is fixed. The set of λ_a for each ANCL approach is as follows: (a) A-EWC: [10, 100, 1000, 10000, 20000, 40000], (b) A-MAS: [1, 5, 10, 50, 100, 200], (c) A-LwF: [0.05, 0.1, 0.5, 1, 5, 10], and (d) A-LFL: [10, 50, 100, 200, 400, 800]

In this section, we measure Euclidean Distance from the weight of ANCL models to the weight of the previous model and the auxiliary model respectively. If the parameter changes less, it is reasonable to expect that less forgetting will happen. According to Mirzadeh et al. [30], forgetting $\mathcal{F}_1$ on task 1 is bounded using Taylor expansion of the loss as follow:

$$\mathcal{F}_1 = \mathcal{L}_1(\hat{\theta}_2) - \mathcal{L}_1(\hat{\theta}_1) \approx \frac{1}{2}(\hat{\theta}_2 - \hat{\theta}_1)^T \nabla^2 \mathcal{L}_1(\hat{\theta}_1)(\hat{\theta}_2 - \hat{\theta}_1) \tag{30}$$

$$\leq \frac{1}{2}\lambda_1^{max} ||\hat{\theta}_2 - \hat{\theta}_1||_2^2 \tag{31}$$

where $\mathcal{L}_1$ is the empirical loss on task 1 and $\nabla^2 \mathcal{L}_1(\hat{\theta}_1)$ is the Hessian for $\mathcal{L}_1$ at $\hat{\theta}_1$. λ_1^{max} is the maximum eigenvalue of $\nabla^2 \mathcal{L}_1(\hat{\theta}_1)$. Above inequality implies that the forgetting $\mathcal{F}_1$ is bounded by the norm of the difference between two weights near the minima of task 1.

Provided that our model learns task t now, the previous model refers to freezed and saved optimal network that has been trained until task $t-1$ and the auxiliary model is the network that is initialized by the weight of the previous model and trained on task t. Then, we can measure two types of weight distance (WD):

$$WD_{old} = ||\theta_t^{ANCL} - \theta^*_{1:t-1}||_2, \qquad WD_{aux} = ||\theta_t^{ANCL} - \theta_t^*||_2. \tag{32}$$

WD_{old} and WD_{aux} calculate the distance from the parameter of ANCL network θ_t^{ANCL} to the parameter of old network $\theta^*_{1:t-1}$ and the parameter of auxiliary network θ_t^* respectively in the same setting as Fig. 3.

The weight distance of ANCL approaches is shown in Fig. 4. We calculate the weight distance with different λ_a in Eqn. (3, 4, 5) which directly adjusting the trade-off between memorizing old tasks and learning new tasks. The model weight remains close to the old weight when λ_a is small, which can be seen in the left side of all figures. In (a) A-EWC and (b) A-MAS, WD_{aux} decreases and WD_{old} increases as λ_a grows larger. This agrees with the analysis of ANCL gradient in Appendix C that the model will converge to the auxiliary network if last term in Eqn. (3) becomes dominant within the loss. In (c) A-LwF and (d) A-LFL, WD_{aux} becomes relatively lower than WD_{old} with increasing λ_a but overall distance become larger for both WD_{old} and WD_{aux}. It is reasonable to think that this is due to the difference between regularization-based methods and distillation-based methods. Unlike

EWC and MAS which directly regularize weight itself, LwF, and LFL choose rather indirect measure, applying loss term based on activation or logit. Thus, in the distillation approaches, the model weight tends to move closer to the auxiliary weight with increasing λ_a but not directly toward it like EWC and MAS.

F.3 Centered Kernel Alignment

Centered Kernel Alignment (CKA) [19] measures the similarity of two representation on the same set of data. Given N data and p neurons, the layer activation matrices $R_1 \in \mathbb{R}^{N\times p}$ and $R_2 \in \mathbb{R}^{N\times p}$ are generated by two layers from two different networks. Then, CKA is defined as:

$$CKA(R_1, R_2) = \frac{HSIC(R_1, R_2)}{\sqrt{HSIC(R_1, R_1)}\sqrt{HSIC(R_2, R_2)}} \tag{33}$$

HSIC stands for Hilbert-Schmidt Independence Criterion [13].We use linear HSIC to implement CKA. Ramasesh et al. [31] measured the layer-wise CKA similarity scores to study catastrophic forgetting on CIFAR datasets. They argue that forgetting happens at deeper layers close to output, as there is a severe drop in CKA score in deeper layers compared to initial layers.

In this analysis, we measure three CKA similarity following training regime in Fig. 3:

$$CKA_{old} = \frac{1}{L}\sum_{l=0}^{L} CKA(R_{t,l}^{ANCL}, R_{1:t-1,l}^{*}), \tag{34}$$

$$CKA_{aux} = \frac{1}{L}\sum_{l=0}^{L} CKA(R_{t,l}^{ANCL}, R_{t,l}^{*}), \tag{35}$$

$$CKA_{multi} = \frac{1}{L}\sum_{l=0}^{L} CKA(R_{t,l}^{ANCL}, R_{t,l}^{multi}). \tag{36}$$

where CKA is calculated over all layers l in resnet32 and averaged. R_t^{ANCL}, $R_{1:t-1}^{*}$ and R_t^{*} are the activation matrices of the ANCL, old, and auxiliary networks, respectively. R_t^{multi} is the activation output from multitask model trained on whole data $D_{1:t}$ until task t. Thus, if CKA_{multi} is high, the model is likely to perform well on all task like multitask model.

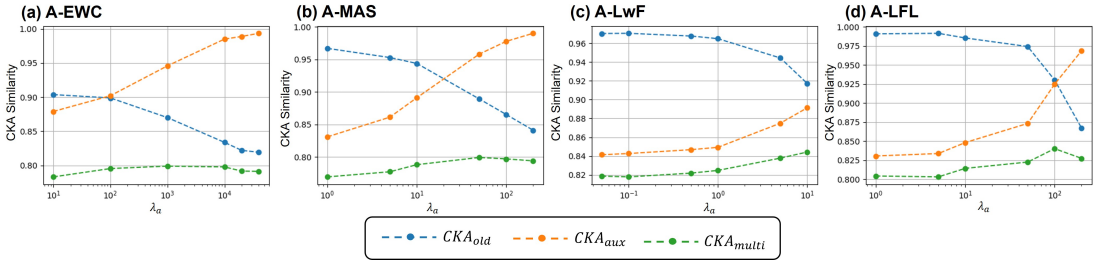

Figure 5: CKA similarity for four ANCL methods on CIFAR-100 (when $t = 2$ in Eqn. (34, 35, 36)). CKA similarity is measured along six different λ_a. The set of λ_a used for each approach is the same as Fig. 4 and λ is fixed during the experiment.

In Fig. 5, three types of CKA (CKA_{old}, CKA_{aux}, and CKA_{multi}) are calculated with different λ_a that adjusts the stability-plasticity balance. CKA_{old}, CKA_{aux}, and CKA_{multi} measures the CKA similarity from ANCL model to old, auxiliary, and multitask model respectively. In all methods, increasing λ_a results in higher CKA_{aux} and lower CKA_{old}, which means that the representation of the ANCL network becomes more similar to that of the auxiliary network and less similar to that of the old network. We can clearly see stability-plasticity trade-off is made by simply controlling λ_a. On the other hand, if CKA_{multi} is high, it can be interpreted that the model tends to generate representation similar to the multitask model, which is the goal of continual learning. Thus, the model is highly likely to be at the best trade-off where λ_a achieve the highest CKA_{multi}. For example, (b) A-MAS and (d) A-LFL reach the best trade-off at $\lambda_a = 50$ and $\lambda_a = 100$ respectively.

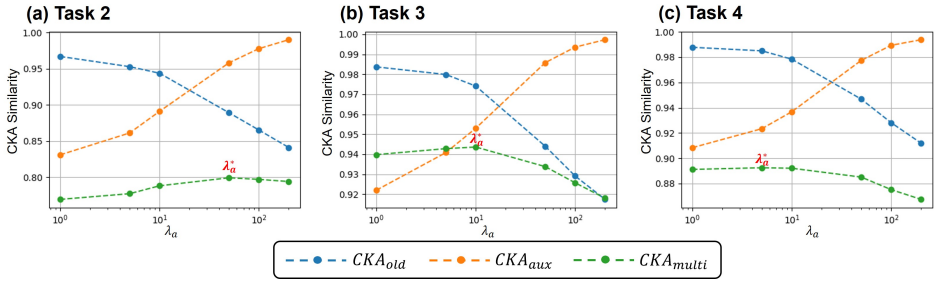

Figure 6: Change in the CKA similarity of A-MAS on task 2, 3, and 4 (when t=2, 3, and 4 in Eqn. (34, 35, 36)). λ_a^* denotes λ_a with the highest CKA_{multi} and is shifted to the left as training progresses.

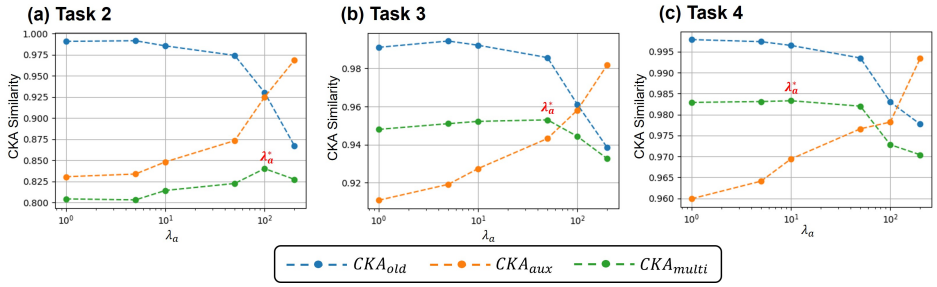

Figure 7: Change in the CKA similarity of A-LFL in tasks 2, 3 and 4 (when t = 2, 3 and 4 in Eqn. (34, 35, 36)). λ_a^* denotes λ_a with the highest CKA_{multi} and is shifted to the left as training progresses.

In addition, one might ask how the best trade-off changes in a sequence of tasks. To answer this question, we also plot CKA figures on different tasks in Fig. 6 and 7. Here, we focus on the optimal λ_a with the highest CKA_{multi} which is denoted as λ_a^* inside the figure. In Fig. 6, λ_a^* changes as 50, 10 and 5 on the task 2-4 and in Fig. 7, λ_a^* changes as 100, 50 and 10. One promising reason for this shrinking λ_a^* is that there are more things to remember as we increase the number of task. In other words, optimal model favors more stability over plasticity in later tasks with respect to the trade-off. This phenomenon is again seen by the analysis of mode connectivity in Fig. 10 and 11, which motivates our new approach of Adap-ANCL afterward in Appendix G.

F.4 Mode Connectivity

Recent works [9], [12] find a simple curves between two local optima of deep neural networks (DNN) such that train loss and test error remain low along these curve. This simple linear path with low error can be visualized in the loss surface of DNN using gradient-based optimization methods. This path called *Mode Connectivity* has been studied empirically and theoretically with some assumptions in different papers.

Venturi et al. [35] theoretically studied mode connectivity when the number of neurons is greater than the number of training data points in one-hidden-layer neural networks. Then, Kuditipudi et al. [21] investigated whether mode connectivity still holds in more realistic setting such as overparameterized networks and noise stable networks. Frankle et al. [10] analyze the effect of SGD noise on gradeint-based neural networks and linearly connected minimum was used to determine the outcome of optimization.

Mode connectivity in continual learning is first studied by Mirzadeh et al. [29]. They empirically showed that the multitask solution and the continual solutions are linear connected in loss landscape assuming that both start from same initialization. We also follow Mirzadeh et al. to visualize the mode connectivity in a two-dimensional plot.

In Fig. 8, we can actually check that continual learning solutions (θ_1^{multi} and θ_2^*) are linearly connected to multitask solution (θ_2^{multi}) in both loss and accuracy landscape. In the left and center columns, multitask weight achieves higher accuracy and low loss compared to continual learning

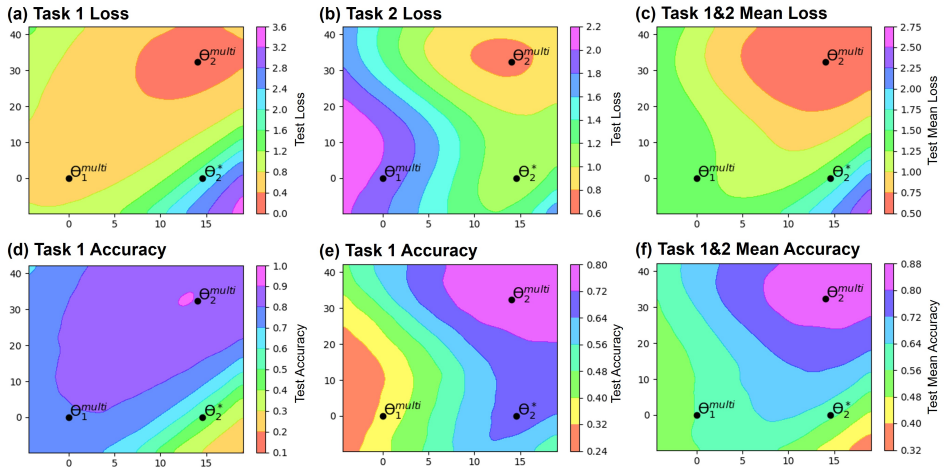

Figure 8: Cross-entropy test loss surface (upper row) and test accuracy surface (lower row) on task 1&2 of CIFAR-100. θ_1^{multi}, θ_2^*, and θ_2^{multi} are used to plot the two-dimensional subspace as Mirzadeh et al. [29].

solutions. This is because every weight is fine-tuned on the dataset and the multitask model has higher discriminating ability due to its full access to previous datasets. In the right column, we visualize the mean loss and mean accuracy of task 1 and 2. Then, it is clear that higher accuracy and lower loss is achieved somewhere in the middle of θ_1^{multi} and θ_2^*. We follow the training scheme in Fig. 3 and θ_1^{multi} is equivalent to the model trained on D_1 only.

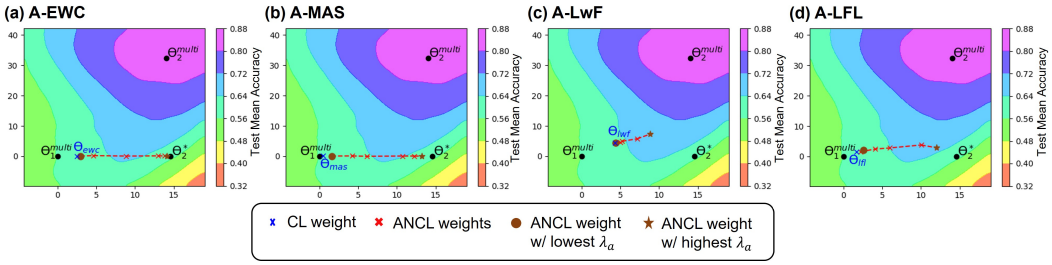

Figure 9: Mean accuracy surface for four ANCL methods on task 1&2 of CIFAR-100. CL and ANCL weights are visualized after projection in the two-dimensional subspace created as Figure 8. As we gradually increase λ_a, the ANCL weight becomes closer to the auxiliary weight θ_2^*. The set of λ_a used in each method is same as the one in Fig. 4.

Next, we project CL (EWC, MAS, LwF and LFL) and ANCL (A-EWC, A-MAS, A-LwF and A-LFL) solutions on the previous graph and picture it as Fig. 9. We change λ_a to see how the ANCL weight changes in the weight space. In (a) A-EWC and (b) A-MAS, it is clearly seen in the figure that λ_a adjusts the interpolation between θ_{CL} and θ_2^*. Large λ_a drift the ANCL weight θ_{ANCL} directly toward the auxiliary weight θ_2^* and the ANCL achieves the highest accuracy on the interpolation as it is in the higher contour. Similarly in (c) A-LwF and (d) A-LFL, the ANCL weight θ_{ANCL} starts near the CL weight θ_{CL} and tends to move toward auxiliary weight θ_2^*. As distillation methods have more flexibility in balancing stability and plasticity, its optimum is able to climb up to the higher contour of mean accuracy than the ANCL applied to regularization-based methods. As a result, the best trade-off is made at somewhere between the old weight and the auxiliary weight. Again we want to emphasize that this is the projection of weight in the two-dimensional space built by three weights (θ_1^{multi}, θ_2^*, and θ_2^{multi}). In other words, it approximates the relative position of CL and ANCL weight but does not show exact position of them in weight space.

Similar to Fig. 6 and 7, we demonstrate how optimal λ_a changes in weight space as training is proceeded. In Fig. 10 and 11, the mean accuracy plot on different tasks is drawn with projected CL

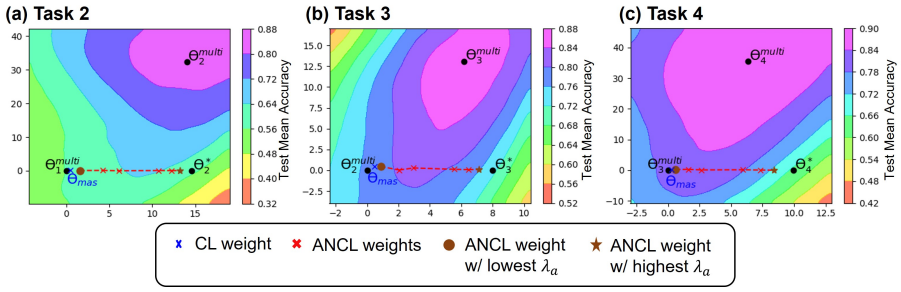

Figure 10: Mean accuracy surface for A-MAS on task 2, task 3, and task 4 on CIFAR-100. Mean accuracy is calculated over all tasks that have been seen so far (i.e., task 1-2 for left column, task 1-3 for center column, and task 1-4 for left column). As training progresses, the contour of high mean accuracy is shifted toward old weight. Thus, the lower value of λ_a is favored in the later task.

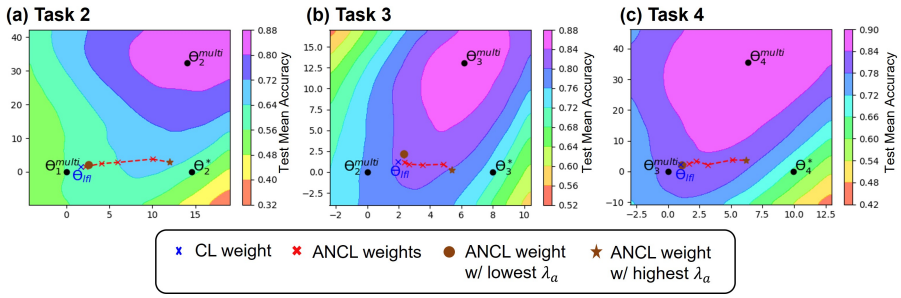

Figure 11: Mean accuracy surface for A-LFL on task 2, task 3, and task 4 on CIFAR-100. Mean accuracy is calculated over all tasks that have been seen so far (i.e., task 1-2 for the left column, task 1-3 for the center column, and task 1-4 for the left column). As training progresses, the contour of high mean accuracy is shifted toward old weight. Thus, the lower value of λ_a is favored in the later task.

and ANCL weight. As we train our model on task 2, 3, and 4, the contour of mean accuracy becomes biased toward the old weight θ_t^{multi}. Thus, λ_a^* with the highest mean accuracy is becoming smaller as the optimal weight prefers the interpolation close to the old weight. This coincides with the results observed in Fig. 6 and 7 that λ_a shrinks as the model learns more tasks. In conclusion, smaller λ_a achieves better trade-off in later tasks. Motivated by these observations, we propose a new technique called *Adaptive ANCL* (Adap-ANCL). Instead of using a fixed independent parameter, Adap-ANCL measures λ_a in a task-dependent manner.

G Adaptive ANCL

ANCL methods solve the stability-plasticity dilemma by adjusting the ratio between λ and λ_a, which is chosen by grid search. However, fixed λ_a can results in worse balance between retaining old knowledge and learning new knowledge because previous knowledge is accumulated as training is proceeded. Thus, it is reasonable to diminish λ_a to put more weight on remembering previous tasks compared to learning a new tasks. We defined our new $\lambda_{a,t}$ on task t as follows:

$$\lambda_{a,t} = \frac{C_t}{C_{1:t}} \lambda_{a,init} \tag{37}$$

C_t denotes the number of classes at task t and $C_{1:t}$ means all classes until task t. $\lambda_{a,init}$ is initial λ_a determined by grid search. Thus, our adaptive parameter $\lambda_{a,t}$ reduces according to the ratio between the number of current classes and the total classes that have been trained so far. In this way, ANCL is able to find a better balance between stability and plasticity in every task.

In Table 1, ANCL with adaptive λ_a gives extra boost accuracy in all methods. This means that adap-ANCL can dynamically find optimal weight by applying flexible hyperparameter.