AdaCL: Adaptive Continual Learning

Elif Ceren Gok Yildirim **Murat Onur Yildirim** **Mert Kilickaya** **Joaquin Vanschoren**
Automated Machine Learning Group, Eindhoven University of Technology

Abstract

Class-Incremental Learning aims to update a deep classifier to learn new categories while maintaining or improving its accuracy on previously observed classes. Common methods to prevent forgetting previously learned classes include regularizing the neural network updates and storing exemplars in memory, which come with hyperparameters such as the learning rate, regularization strength, or the number of exemplars. However, these hyperparameters are usually only tuned at the start and then kept fixed throughout the learning sessions, ignoring the fact that newly encountered tasks may have varying levels of novelty or difficulty. This study investigates the necessity of hyperparameter 'adaptivity' in Class-Incremental Learning: the ability to dynamically adjust hyperparameters such as the learning rate, regularization strength, and memory size according to the properties of the new task at hand. We propose AdaCL, a Bayesian Optimization-based approach to automatically and efficiently determine the optimal values for those parameters with each learning task. We show that adapting hyperpararmeters on each new task leads to improvement in accuracy, forgetting and memory. Code is available at `https://github.com/ElifCerenGokYildirim/AdaCL`.

1 Introduction

This paper focuses on Class-Incremental Learning of deep neural network representations (Masana et al., 2020; De Lange et al., 2021). Unlike standard batch learning, which requires access to data from all categories simultaneously, Class-Incremental Learning can update a pre-trained deep classifier with new categories by expanding the classifier layer with new output nodes for new classes. This leads to more efficient learning and avoids the need to store task identities which often are not available in real-world scenarios.

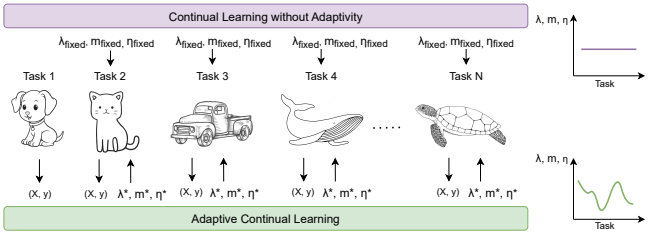


Figure 1: Comparison of fixed *vs.* adaptive continual learning (AdaCL). In this work, we hypothesize that different tasks may require different settings and explore the potential of tuning learning rate (η), regularization strength (λ) and memory size per task (m), allowing to learn adaptively.

While Class-Incremental Learning enables expanding a classifier without requiring task identities, it often results in *catastrophic forgetting*. This occurs when the deep learner sacrifices accuracy on previously seen classes to learn new ones. Three major approaches have been explored to address this issue: regularization, replay and architecture adaptation. Regularization prevents abrupt shifts in the neural network weights while learning new classes (Kirkpatrick et al., 2017; Li and Hoiem, 2017). Replay stores a few exemplars per class in memory and replays them during new learning increments (Lopez-Paz and Ranzato, 2017). Architecture-based approaches build network structures by either expanding the existing network (Rusu et al., 2016; Yan et al., 2021) or by partially isolating network parameters to retain past class information (Liu et al., 2021a; Kang et al., 2022; Dekhovich et al., 2023). Although these methods improve the performance, they always use a fixed learning rate, regularization magnitude, and pre-defined memory size throughout the learning process, which is likely suboptimal.

This paper addresses the issue of dynamically adjusting *how much* to regularize or store in memory for each new task. We explore whether adaptation is necessary for optimal performance, treating the learning rate, regularization magnitude, and memory size as latent variables that should

Proceedings of the 1st ContinualAI Unconference, 2023, Virtual. PMLR Volume 249. Copyright 2023 by the author(s).

be adjusted based on the current state of the learner and the complexity of the task (see Figure 1). We use Bayesian Optimization to efficiently discover the best hyperparameters per task. Our experiments on CIFAR-100 and MiniImageNet demonstrate that adapting these parameters to the tasks results in significant improvements and give us new insight into how to adapt to various new tasks. In summary, this paper makes the following contributions:

I. In this paper, for the first time, we raise the important issue of adaptive hyperparameter selection in class-incremental learning.

II. We propose to predict the learning rate, regularization magnitude, and memory size conditioned on the state of the deep learner and the current learning task via Bayesian Optimization.

III. Through large-scale experiments on well-established benchmarks, we show that learning adaptively yields significant performance and efficiency improvements, both increasing accuracy and reducing forgetting.

2 Related Work

Class-Incremental Learning. Class-Incremental Learning updates a deep classifier with sequentially arriving data, usually with mutually exclusive categories (Masana et al., 2020; De Lange et al., 2021; Wang et al., 2023; Zhou et al., 2023; Kilickaya et al., 2023). However, when novel data arrives, previous training data becomes unavailable, leading to catastrophic forgetting. To mitigate this, researchers have developed three main approaches: (i) regularization-based methods, which stabilize important parameters or distill previous knowledge into the model (Kirkpatrick et al., 2017; Zenke et al., 2017; Lee et al., 2017; Li and Hoiem, 2017; Chaudhry et al., 2018a; Zhou et al., 2021b; Zhu et al., 2021), (ii) replay-based methods, which usually benefit from regularization-based methods and store a subset of training data to rehearse during learning (Rebuffi et al., 2017; Chaudhry et al., 2018b; Wu et al., 2019; Aljundi et al., 2019; Ostapenko et al., 2019; Xiang et al., 2019; Zhao et al., 2020; Liu et al., 2021b; Petit et al., 2023) and (iii) architecture-based methods redesign network architectures by extending the network (Rusu et al., 2016; Yan et al., 2021; Zhu et al., 2022) or freezing network parameters partially to preserve old class knowledge (Liu et al., 2021a; Kang et al., 2022; Dekhovich et al., 2023).

However, current studies assume a constant amount of regularization and memory size per task throughout learning sessions, which is unnatural since learning unfamiliar objects requires more plasticity than learning familiar ones (Cha and Cho, 2024). To address this issue, we propose an adaptive method in which the regularization magnitude, learning rate and memory size are automatically tuned within each incremental learning step.

Hyperparameter Optimization. Hyperparameter Optimization (HPO) aims to optimize the hyperparameters of a given deep learning model, including the learning rate, layer size, or balance of different loss functions. In this paper, our focus is on balancing the contribution of a standard cross-entropy and regularization loss, learning rate as well as memory size per task if applicable. To tackle the HPO problem, complex techniques such as bi-level optimization (Franceschi et al., 2018) or gradient-based optimization (Baydin et al., 2018) have been proposed. Bi-level optimizers alternate between optimizing neural network weights and tuning the hyper-parameters, while gradient-based methods treat all network weights as hyperparameters to be updated.

Several recent studies (Chaudhry et al., 2019; De Lange et al., 2021; Liu et al., 2023) share our core motivation by investigating the impact of hyperparameter optimization in subsequent tasks. De Lange et al. (2021) adopt a two-stage strategy: First, they fine-tune the current task to identify the optimal learning rate with a grid search for maximum plasticity and peak accuracy. Second, they introduce a new thresholding hyperparameter to naively balance the plasticity and stability trade-off: starting with a high regularization strength and decaying it when the performance of the current task is below the defined threshold. However, this approach follows a very naive search since they basically apply two consecutive grid searches to decide the optimal value. Moreover, they focus on a Task-Incremental setup and do not consider the memory size in their search space.

Chaudhry et al. (2019) tunes the hyperparameters for the first T tasks with a grid search and then uses the best-found values in the remaining tasks. However, it assumes that the initial few tasks are representative enough for the rest of the tasks which may not be realistic in most of the cases. Again, they worked on the Task-Incremental scenario and did not consider the memory size in their search space.

Liu et al. (2023) uses reinforcement learning in a Class-Incremental scenario to adaptively find the best hyperparameter values while learning the tasks. They hold a validation set, similar to our study, to estimate rewards by finding the best set of hyperparameters. However, its search space is limited to learning rate, regularization strength, and the type of classifier.

In this work, we propose Bayesian Optimization (Snoek et al., 2012) with Tree Parzen Estimators due to its effectiveness over multiple hyperparameters. We evaluate the generality of our approach by dynamically tuning the learning rate, regularization strength, and memory size across a stream of tasks.

3 Method

Overview. Class-incremental learning involves updating a neural network with new classes as it comes in. Specifically, the learner receives a sequence of learning tasks $\mathcal{T}_{1:t} = (\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_t)$, each with a corresponding dataset $\mathcal{D}_{\mathcal{T}} = (x_{i,t}, y_{i,t})^{n_t}$ consisting of n_t instances per task. Each input pair $x_{i,t}, y_{i,t} \in \mathcal{X}_t \times \mathcal{Y}_t$ is sampled from an unknown distribution where $x_{i,t}$ is the sample and $y_{i,t}$ is the corresponding label. It's important to note that the learning tasks are mutually exclusive, i.e., $\mathcal{Y}_{t-1} \cap \mathcal{Y}_t = \emptyset$. When a new learning task arrives, the deep convolutional network is optimized to embed the input instance into the classifier space $f_\Theta : \mathcal{X}_t \rightarrow \mathcal{Y}_t$, where Θ represents the parameters of the learner.

The incremental learner has two goals: to effectively learn the current task (*plasticity*) while retaining performance on all previous tasks (*stability*). This can be accomplished by optimizing the following function where $CE(\cdot)$ represents the Cross-Entropy used in classification, and $Reg(\cdot)$ is a regularization term that penalizes abrupt changes in the neural network weights (Li and Hoiem, 2017; Kirkpatrick et al., 2017; Rebuffi et al., 2017; Zhao et al., 2020):

$$\mathcal{L} = CE(f(x_{i,t}), y_{i,t}) + \lambda \cdot Reg(\Theta) \tag{1}$$

3.1 Base Models for AdaCL

AdaCL can be combined with many base incremental learners. We experimented with four popular, well-established techniques: EWC (Kirkpatrick et al., 2017), LwF (Li and Hoiem, 2017) iCaRL (Rebuffi et al., 2017) and WA (Zhao et al., 2020). We select baselines that complement each other and serve as strong baselines within the field of incremental learning (Table 1).

Table 1: Selected models to evaluate the impact of adaptivity in Class-Incremental Learning.

method	prior-based	distillation-based	exemplar collection	classifier correction
EWC	✓			
LwF		✓		
iCaRL		✓	✓	
WA		✓	✓	✓

EWC. Elastic Weight Consolidation (Kirkpatrick et al., 2017) is a weighted regularization approach. The authors argue that not all weights contribute equally to learning a new task and estimate the importance of each weight in minimizing the classification loss for the current task: $Reg(\Theta) = ||\mathcal{F}(\Theta - \Theta')||$, where Θ' is the model weights from the previous learning step, $\mathcal{F}$ is the Fisher matrix of the same size as the weight matrices Θ, re-weighting the contributions of each weight to stabilize the important neurons per task.

LwF. Learning-without-Forgetting (Li and Hoiem, 2017) is a knowledge-distillation approach where the teacher branch is the model from the previous task, and the student branch is the current model. The aim is to match the activations of the teacher and student branches, either at the feature or logit level. Formally, LwF minimizes the following objective where f' is the model from the previous learning step, and $KL(p_1, p_2)$ is the KL-divergence between two probability distributions p_1 and p_2:

$$Reg(\Theta) = KL(f(x_{i,t}), f'(x_{i,t})) \tag{2}$$

iCaRL. The Incremental Classifier and Representation Learning (Rebuffi et al., 2017) leverages a hybrid approach that involves two main components: exemplar-based memory which is carefully selected to maintain representation and a regularization. The exemplar-based memory module retains a subset of exemplar samples from previous tasks, representing important instances that encapsulate the learned knowledge. By utilizing exemplars, iCaRL ensures the model's ability to recognize and classify past instances while discriminating between learned and new classes. The distillation loss as in Eq. 2 used for regularization, enables knowledge distillation from previous models to guide the learning process for new tasks. This distillation process allows the model to align logits of new classes with already learned classes to mitigate catastrophic forgetting.

WA. Maintaining Discrimination and Fairness in Class Incremental Learning (Zhao et al., 2020) is a method that consists of two phases: maintaining discrimination and maintaining fairness. The first phase is similar to the previously established method (Rebuffi et al., 2017). Their study demonstrates that knowledge distillation is not sufficient by itself to prevent the model to treat old classes and new classes fairly since there is a high tendency towards new classes in the classifier layer to minimize the Eq 2. Therefore, the second stage named Weight Aligning (WA) focuses on maintaining fairness to correct this classifier bias towards new classes. WA showed that it treats all classes fairly, and significantly improves the overall performance.

3.2 Constancy Assumption in Class Incremental Learning

The scalar parameter λ balances the contribution of the classification and regularization loss functions. A large value of λ ensures minimal weight updates, which can sacrifice learning on the current task. Conversely, a small λ yields good performance on the current task but may sacrifice performance on previous tasks, exacerbating catastrophic forgetting. Similarly, requirement for a fixed or predetermined memory size per task may not always be optimal, as it depends on the new task and its relationship to previous tasks. Specifically,where the new task is highly

similar to previous tasks, it is possible to retain past knowledge by storing only a small number of representative samples. Conversely, when the new task is significantly distinct, it is reasonable to store a larger number of examples in memory to prevent catastrophic forgetting while learning new tasks. However, as a common practice, important hyperparameters such as learning rate (η), regularization strength (λ), and memory size (m) are set to a fixed or pre-defined scalar value throughout all incremental learning sessions with $t \in \mathcal{T}_{1:t}$; such that $\eta_t = \eta_{t-1}$, $\lambda_t = \lambda_{t-1}$ or $\lambda_t = \frac{t*c}{(t*c)+c}$ where c is the number of classes per task. Similarly, $m_t = m_{t-1}$ or $m_t = \frac{M}{t}$ where M is the pre-defined total memory size.

We hypothesize that the assumption of *constant or pre-defined learning rate, regularization strength, and exemplar size per task* is suboptimal for building accurate lifelong learning machines. Our reasoning is two-fold:

Low Plasticity and High Stability. The incremental learner may encounter a novel object that is highly familiar with the previously learned tasks. For example, it may encounter the category *dog* after observing many other animal categories, such as *cat, cow, bird*. In this case, the learner does not need to store many exemplars from previous tasks or to be too plastic, as it can quickly transfer knowledge from the previous tasks where it is similar to the human learning process and referred to *low road transfer* (Perkins and Salomon, 1992). Hence, no drastic updates to the learned filters are necessary.

High Plasticity and Low Stability. Conversely, the learner may encounter a novel object that is highly unfamiliar with the previous tasks. For example, it may encounter the category *car* after observing many other animal categories, such as *cat, cow, bird*. In this case, the learner would require replaying more exemplars from previous tasks to preserve old knowledge and high plasticity to learn about the novel object with never-before-seen parts, such as wheels.

3.3 AdaCL: Adaptive Continual Learning

AdaCL aims to optimize the regularization magnitude λ and memory size m as a function of model performance over a set of incremental tasks, conditioned on the current learning task and all previous tasks. We define $\eta(t) = \eta_1, \eta_2, \ldots, \eta_{t-1}, \eta_t$, and $\lambda(t) = \lambda_1, \lambda_2, \ldots, \lambda_{t-1}, \lambda_t$, and $m(t) = m_1, m_2, \ldots, m_{t-1}, m_t$ where η_t, λ_t and m_t are predicted by minimizing the following optimization problem:

$$\arg\min_{\eta,\lambda,m} \mathcal{L}(\Theta; V_t) = \arg\min_{\eta,\lambda,m} \sum_{i=1}^{|V_t|} [CE(f(x_{i,t}; \Theta), y_{i,t}) \quad (3)$$

Here, V_t is a randomly selected class-balanced subset of the current task and previous tasks that guide the model's adaptation with careful consideration of both new and previous tasks' characteristics and prevents bias over certain classes. $\mathcal{L}(\Theta; V_t)$ is the loss function where the learning rate η, the regularization coefficient λ, and memory size per task m is determined by solving the optimization problem. Our adaptive approach, AdaCL (Algorithm 1), starts after the first task since it is just a standard batch learning.

Algorithm 1 AdaCL: Adaptive Continual Learning

Require:
θ_{t-1} ▷ model from previous task
X_t ▷ dataset from new task
$M_{t-1} = m_1, \ldots, m_{t-2}, m_{t-1}$ ▷ memory from old tasks
$V_{t-1} = v_1, \ldots, v_{t-2}, v_{t-1}$ ▷ val. set from tasks seen so far
η_{space} ▷ search space for learning rate
λ_{space} ▷ search space for regularization
m_{space} ▷ search space for memory
$configs, epochs$ ▷ # of configurations and epochs
$V_t = V_{t-1} \cup v_t \leftarrow X_t$
for $c = 1, \ldots, configs$ **do**
  $\eta_t \leftarrow \eta_{space}$ ▷ η for new task
  $\lambda_t \leftarrow \lambda_{space}$ ▷ λ for new task
  $M_t : m_t \leftarrow m_{space}$ ▷ memory with a size of m_t
  $D = X_t \cup M_{t-1} \cup M_t$ ▷ concat new data and memory
  for $e = 1, \ldots, epochs$ **do**
    Train Eq. 1 with θ_{t-1} and D
    Evaluate Eq. 3 with V_t
  end for
end for
return $\theta_t, V_t, \eta_t^*, \lambda_t^*, M_t^*$ ▷ new model with optimal learning rate, regularization strength and memory size

In the following tasks, it retains the model θ_{t-1} trained on the previous task, receives current task data X_t, and creates a validation set V_t. Then, training data D is constructed and trained with Eq. 1 after the configuration for η_t, λ_t and m_t is selected by Bayesian Optimization (see Section 3.4). After each epoch, the selected configuration is evaluated on the validation set V_t with Eq. 3. Subsequently, this process is repeated until reaching the total number of configurations. The optimal learning rate η_t^*, lambda λ_t^*, and memory size per task m_t^* are determined based on the validation performance.

This approach allows us to automatically adjust the learning rate, regularization strength, and memory size per task according to the specific learning task based on the given loss function which lets the model find the degree of difficulty itself, avoiding the unrealistic assumption of a fixed learning rate, regularization strength, and memory size throughout the learning process.

3.4 Bayesian Optimization via Parzen Estimator

We optimize the objective function using multivariate tree-structured parzen estimators (TPE) (Bergstra et al., 2011). TPE builds a conditional probability tree that maps hyperparameters to their respective model performances. Then it can be used to guide a search algorithm to find the optimal set of hyperparameters for the given model. In this study, TPE is utilized as a search algorithm where it searches

within the provided range for learning rate, regularization strength, and memory size per task and then searches for the best value by evaluating across accumulated validation set which consists of previous and new tasks throughout incremental learning sessions. We use Optuna (Akiba et al., 2019) for TPE implementation.

4 Experimental Protocol

In this section, we describe our experimental setup, present our findings and results, and provide an ablation study. **Datasets.** In this paper, we experiment with **CIFAR100** (Krizhevsky et al., 2009) and **MiniImageNet** (Vinyals et al., 2016). Each dataset exhibits objects from 100 different categories. We train all the models with 10 tasks, with 10 classes within each learning task on both CIFAR100 and MiniImageNet. Both datasets have 5000 training, and 1000 testing color images per learning task, each with 32×32 and 64×64 resolution for CIFAR100 and MiniImageNet respectively.

Metrics. We resort to the standard metrics for evaluation, accuracy (ACC) which measures the final accuracy averaged over all tasks, and backward transfer (BWT) which measures the average accuracy change of each task after learning new tasks:

$$ACC = \frac{1}{T}\sum_{i=1}^{T} A_{T,i} \tag{4}$$

$$BWT = \frac{1}{T-1}\sum_{i=1}^{T-1}(A_{T,i} - A_{i,i}) \tag{5}$$

where $A_{T,i}$ represents the testing accuracy of task T after learning task i.

Baselines. EWC, LwF, iCARL, and WA are our direct baselines since we use them as base models in AdaCL. We also compare our common baseline results with OMDP (Liu et al., 2023). Finally, we select one recent memory-free approach FeTrIL (Petit et al., 2023), and one recent memory-based method PODNet (Douillard et al., 2020) to provide more comprehensive insights.

Implementation Details. We employ adaptive hyperparameter optimization on the methods discussed in section 3.2, and compare them with their fixed (original) versions. For the fixed versions, we use the default η, λ and m as defined in PYCIL (Zhou et al., 2021a).

We use ResNet-32 as the backbone (He et al., 2016). We set the number of epochs to 100 but use the Successive Halving (Li et al., 2018) scheduler for a more efficient search. We use SGD optimizer with momentum parameter set to 0.9 and weight decay $5e^{-4}$ for the first task and $2e^{-4}$ for the rest of the tasks. The batch size is set to 128. We run experiments on three different seeds and report their average.

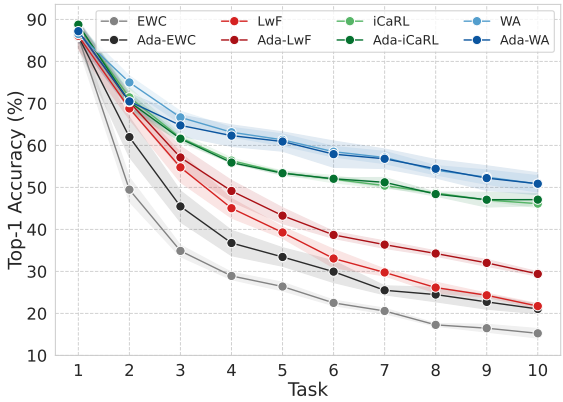


Figure 2: Accuracy after each task on **CIFAR100**. AdaCL significantly boosts the performance on regularization-based methods and improves the efficiency by storing fewer exemplars on memory-based methods while yielding on par performance.

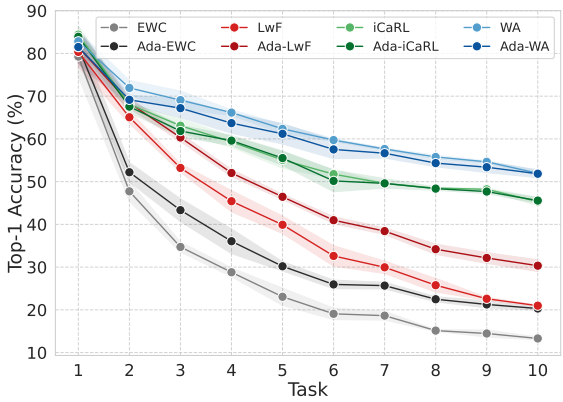


Figure 3: Accuracy after each task on **MiniImageNet**. The results align with the observations on CIFAR100.

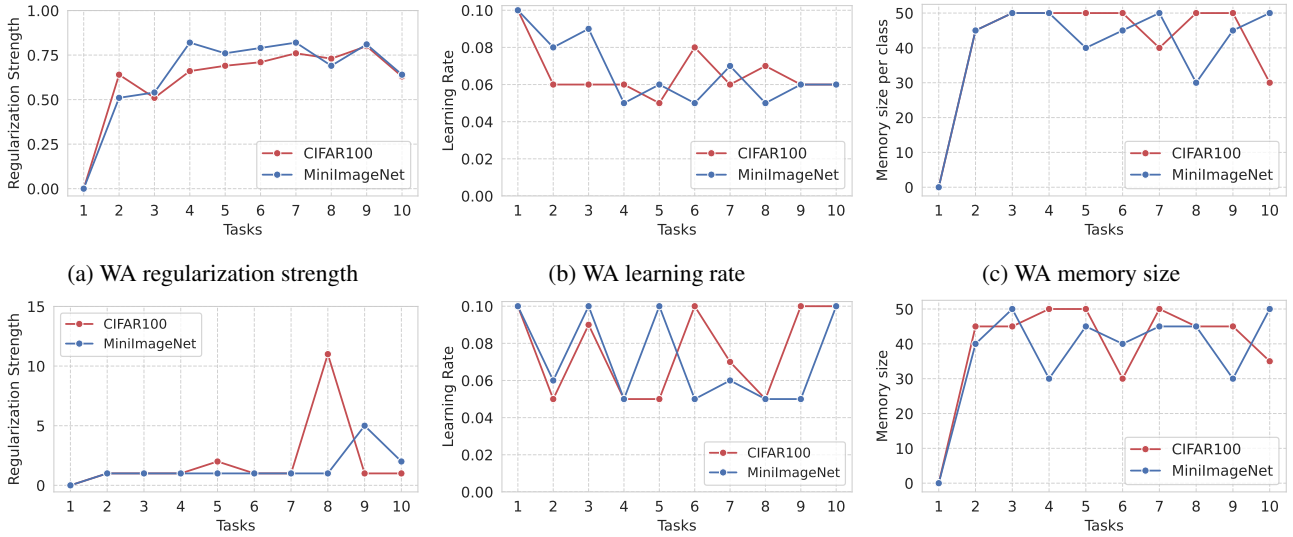
We store a small subset of the validation data from each incremental learning step to evaluate the search algorithm. The search space for the learning rate and the maximum memory size per class within a task is set to [0.05, 0.1] and 50 respectively. The search space for λ is determined based on the ablation experiments and details are given in Appendix A.1.

5 Experimental Results

The Effect of Adaptivity. We investigate the efficacy of our adaptive method compared to traditional fixed hyperparameter approaches across the CIFAR100 and MiniImageNet datasets. Our results highlight significant advancements of our adaptive approach AdaCL, particularly notable in regularization-centric techniques like EWC and LwF as illustrated in Figure 2 and Figure 3.

Table 2: Performance comparison of various methods on the CIFAR100 and MiniImageNet datasets in terms of ACC, BWT, and memory size. Baseline methods such as EWC and LwF do not utilize memory. Our proposed methods, denoted with (ours), demonstrate better or competitive performance across both datasets while using less memory.

	CIFAR100			**MiniImageNet**		
Method	ACC (%)	BWT (%)	Memory Size	ACC (%)	BWT (%)	Memory Size
PODNet	39.47 ± 1.39	-24.14 ± 5.49	4500	43.49 ± 0.31	-10.84 ± 10.83	4500
OMDP	46.94 ± 2.11	-28.34 ± 1.28	4500	46.15 ± 0.55	-25.54 ± 4.18	4500
FeTrIL	27.56 ± 1.50	-18.92 ± 5.70	-	24.46 ± 1.60	-15.84 ± 0.73	-
EWC	15.26 ± 1.37	-63.96 ± 3.21	-	13.30 ± 0.38	-60.64 ± 3.11	-
Ada-EWC (ours)	21.06 ± 1.37	-13.28 ± 3.09	-	20.31 ± 0.39	-11.86 ± 2.32	-
LwF	21.74 ± 0.73	-48.88 ± 12.43	-	20.97 ± 0.19	-50.06 ± 9.59	-
Ada-LwF (ours)	29.41 ± 0.65	-22.34 ± 4.18	-	30.33 ± 1.65	-28.71 ± 6.44	-
iCaRL	46.13 ± 1.35	-28.84 ± 5.06	4500	45.83 ± 1.43	-27.33 ± 5.54	4500
Ada-iCaRL (ours)	46.44 ± 2.50	-28.62 ± 2.85	4125	46.10 ± 2.26	-28.77 ± 4.26	3950
WA	50.84 ± 2.37	-17.23 ± 1.51	4500	51.96 ± 0.74	-22.46 ± 1.67	4500
Ada-WA (ours)	50.87 ± 3.19	-20.49 ± 2.88	4085	51.85 ± 1.12	-24.22 ± 4.41	4050



(a) WA regularization strength (b) WA learning rate (c) WA memory size

(d) iCaRL regularization strength (e) iCaRL learning rate (f) iCaRL memory size

Figure 4: Adaptive modifications in regularization strength, learning rate, and memory allocation. The selected hyperparameters diversely change across task sequences, datasets, and methods and indicate the necessity of adaptivity in CL.

For example, we find 8% and 10% increase in accuracy while 26% and 21% improvement in backward transfer on CIFAR100 and MiniImagenet respectively with LwF by adjusting the regularization strength and learning rate.

In memory-centric methods, we find that they show greater resilience to the changes in hyperparameters. Storing sufficient exemplars aids the model in capturing the distribution of different tasks simultaneously, thereby reducing the reliance on hyperparameter optimization. Despite minor differences, we consistently observe similar accuracy and backward transfer, as seen in Table 2.

Comparison with Recent Baselines. We include results from recent baselines PODNet and FeTrIL to provide comprehensive insights. An intriguing finding is that the initial performance of fixed LwF is outperformed by the recent FeTrIL method, but when we tune LwF, it outperforms FeTrIL by 2% and 6% on CIFAR100 and MiniImageNet, respectively. Furthermore, we compare AdaCL with another HPO-based method OMDP, on iCaRL, our only common baseline. Our performance closely aligns with OMDP but a key advantage of our adaptive approach is that it does so while using less memory.

Memory Allocation. We investigate the memory allocation of iCaRL and WA by specifically tuning the memory size for these methods. We observe that in our adaptive approach we were able to attain similar results while utilizing less memory compared to those obtained from fixed versions as given in Table 2. This is due to AdaCL capability to choose exemplars from both decision boundaries and the center (Figure 5), highlighting how the adaptive approach can achieve comparable results.

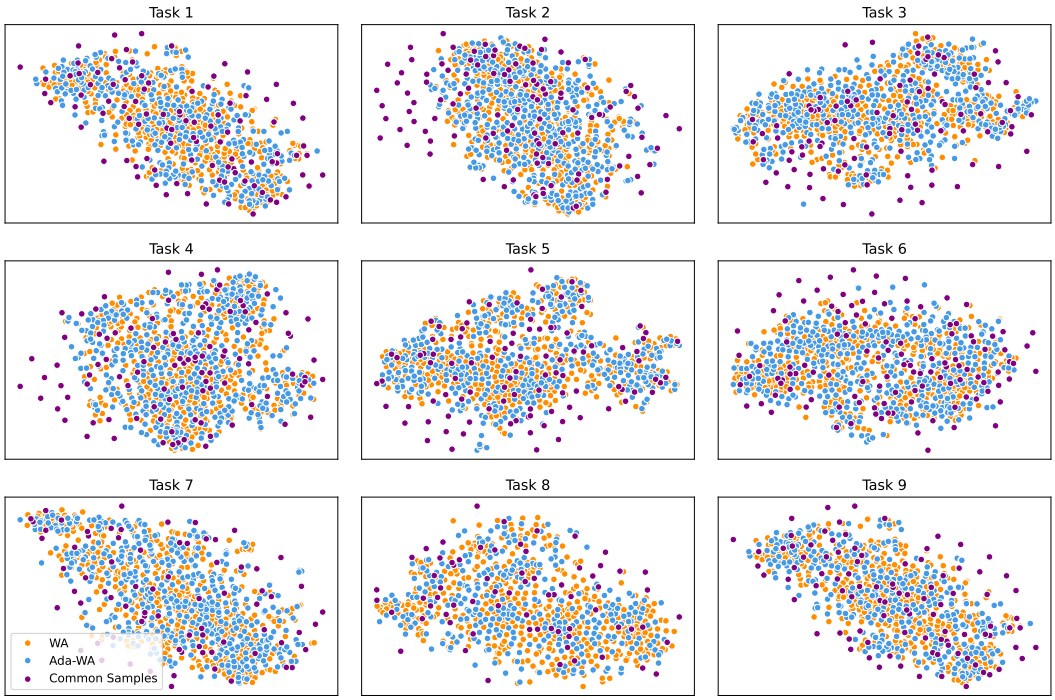


Figure 5: t-SNE plots of selected exemplars. Ada-WA selects exemplars from boundaries and center. This way, it is able to achieve on-par performance with less memory. The final task is omitted from the visualization, since memory selection is not necessary for it.

Exploring Hyperparameter Dynamics. We observe the selected hyperparameters throughout the process of continual learning and reveal intriguing dynamics in the adjustment of regularization strength, learning rate, and memory size across tasks. This observation highlights the crucial role of adaptability in continual learners, allowing them to adapt dynamically to the changing demands of each task.

For example, Figure 4 presents the chosen hyperparameters for WA and iCaRL, illustrating the subtle adjustments made throughout the learning. Please refer to Appendix A.2 for other methods.

5.1 Ablation Study

In Table 3, we provide a comprehensive analysis of how different hyperparameters interact with model performance. Our findings reveal that the performance of the model is intricately linked to the interplay of various hyperparameters. While optimizing solely the learning rate and regularization strength yields the highest accuracy, we also incorporate with memory size to enhance memory allocation efficiency.

Our comprehensive hyperparameter optimization strategy showcases an enhancement in memory efficiency by minimizing the amount of stored exemplars while maintaining on-par accuracy. This insight underscores the importance of not only optimizing individual hyperparameters but also understanding their collective impact on model performance, particularly in scenarios where resource constraints necessitate efficient memory allocation in practical applications.

Table 3: The findings of our ablation study on the WA method, where different hyperparameters were tuned simultaneously and individually on CIFAR100 dataset. Tuning all hyperparameters simultaneously results in a negligible decrease in ACC but yields improvements in memory.

Regularization Strength	Learning Rate	Memory Size	ACC (%)	Stored Memory Size
-	-	-	53.48	4500
✓	-	-	54.00	4500
-	✓	-	53.65	4500
-	-	✓	51.02	3350
✓	✓	-	54.24	4500
✓	-	✓	51.73	3350
-	✓	✓	51.40	3750
✓	✓	✓	53.16	4250

Finally, in Table 4, we investigate the impact of employing random memory selection as an alternative to herding under the WA method. Although there is a slight tendency to decrease in both incremental accuracy and forgetting, we found that the adoption of random memory selection leads to an insignificant change in accuracy and forgetting.

Table 4: An investigation into the WA method when using random and herding as the memory selection strategies on CIFAR100 dataset. Findings indicate that random memory selection produces similar results to herding.

Method	selector	ACC (%)	BWT (%)
WA	herding	50.84 ± 2.37	-17.23 ± 1.51
Ada-WA	herding	50.87 ± 3.19	-20.49 ± 2.88
WA	random	49.95 ± 2.72	-17.59 ± 2.22
Ada-WA	random	49.96 ± 2.55	-21.51 ± 1.28

6 Conclusion

This study introduces the idea of adaptive hyperparameter tuning for Class-Incremental Learning. These hyperparameters are treated as tunable variables that can be adjusted for an each new task according to the learner's current condition and the complexity of the task. Leveraging the sample-efficiency of Bayesian Optimization, the paper presents a methodology to predict the optimal values for these hyperparameters in each learning task. By conducting experiments on well-established benchmarks, the study showcases the remarkable enhancements in performance achieved through adaptive learning, resulting in improved accuracy, diminished forgetting, and less memory. Potential avenues for future investigation could involve the reduction of hyperparameter tuning costs (e.g. via warm-starting) and the exploration of alternative methods for constructing or optimizing the validation set.

To sum up, our study leads the way in introducing the concept of adaptive hyperparameter optimization in Class-Incremental Learning, with a mindful consideration of the limitations we've recognized. As the field further advances, we anticipate that these insights will shape the evolution of advanced continual learning approaches, empowering deep neural networks to adapt to streams of real-world tasks.

Acknowledgements

This work is supported by; TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under GA No. 952215, and the Dutch national e-infrastructure with the support of SURF, Cooperative using grant no. EINF-4569, and the Turkish MoNE scholarship.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. *NeurIPS*.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *JMLR*.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *NeurIPS*.

Cha, S. and Cho, K. (2024). Hyperparameters in continual learning: a reality check. *arXiv preprint arXiv:2403.09066*.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. (2018a). Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2018b). Efficient lifelong learning with a-gem. *arXiv preprint*.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with a-gem. In *ICLR*.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Dekhovich, A., Tax, D. M., Sluiter, M. H., and Bessa, M. A. (2023). Continual prune-and-select: class-incremental learning with specialized subnetworks. *Applied Intelligence*, pages 1–16.

Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 86–102. Springer.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. (2018). Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Kang, H., Mina, R. J. L., Madjid, S. R. H., Yoon, J., Hasegawa-Johnson, M., Hwang, S. J., and Yoo, C. D. (2022). Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*, pages 10734–10750. PMLR.

Kilickaya, M., Van der Weijer, J., and Asano, Y. (2023). Towards label-efficient incremental learning: A survey. *arXiv preprint*.

Kirkpatrick, J., Pascanu, R., et al. (2017). Overcoming catastrophic forgetting in neural networks. *PNAS*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Toronto, ON, Canada*.

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. (2017). Overcoming catastrophic forgetting by incremental moment matching. *NeurIPS*.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., and Talwalkar, A. (2018). Massively parallel hyperparameter tuning. *arXiv preprint arXiv:1810.05934*.

Li, Z. and Hoiem, D. (2017). Learning without forgetting. *TPAMI*.

Liu, Y., Li, Y., Schiele, B., and Sun, Q. (2023). Online hyperparameter optimization for class-incremental learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8906–8913.

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

Liu, Y., Schiele, B., and Sun, Q. (2021b). Rmm: Reinforced memory management for class-incremental learning. *Advances in Neural Information Processing Systems*, 34:3478–3490.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *NeurIPS*.

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

Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., and Nabi, M. (2019). Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*.

Perkins, D. N. and Salomon, G. (1992). Transfer of learning. *International encyclopedia of education*, 2:6452–6457.

Petit, G., Popescu, A., Schindler, H., Picard, D., and Delezoide, B. (2023). Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3911–3920.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *CVPR*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *NeurIPS 2012*, 25.

Vinyals, O., Blundell, C., et al. (2016). Matching networks for one shot learning. NeurIPS'16, page 3637–3645, Red Hook, NY, USA. Curran Associates Inc.

Wang, L., Xingxing, Z., Hang, S., and Jun, Z. (2023). A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint*.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019). Large scale incremental learning. In *CVPR*.

Xiang, Y., Fu, Y., Ji, P., and Huang, H. (2019). Incremental learning using conditional adversarial networks. In *ICCV*.

Yan, S., Xie, J., and He, X. (2021). Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *ICML*.

Zhao, B., Xiao, X., Gan, G., Zhang, B., and Xia, S.-T. (2020). Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13208–13217.

Zhou, D.-W., Wang, F.-Y., Ye, H.-J., and Zhan, D.-C. (2021a). Pycil: A python toolbox for class-incremental learning. *arXiv preprint arXiv:2112.12533*.

Zhou, D.-W., Wang, Q.-W., Qi, Z.-H., Ye, H.-J., Zhan, D.-C., and Liu, Z. (2023). Deep class-incremental learning: A survey. *arXiv preprint*.

Zhou, D.-W., Ye, H.-J., and Zhan, D.-C. (2021b). Co-transport for class-incremental learning. In *ACM MM*.

Zhu, F., Zhang, X.-Y., Wang, C., Yin, F., and Liu, C.-L. (2021). Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880.

Zhu, K., Zhai, W., Cao, Y., Luo, J., and Zha, Z.-J. (2022). Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9296–9305.

A Appendix

A.1 Search Space Range

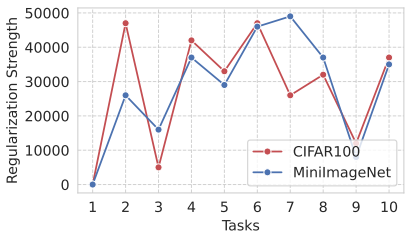
In this section, we present the findings of our search space ablation study conducted to determine the optimal range of regularization strength. Our analysis revealed that regularization strengths within the interval of [1, 100] yielded optimal performance for LwF, thus it is used in our main experiments. Similarly, for EWC the optimal regularization strength falls within the range of [1, 50000] and is employed in our main experiments.

Table 5: Ablation of the search space for regularization hyperparameter on EWC and LwF methods. The search space intervals were determined based on achieving optimal accuracy results. Best results are highlighted in bold.

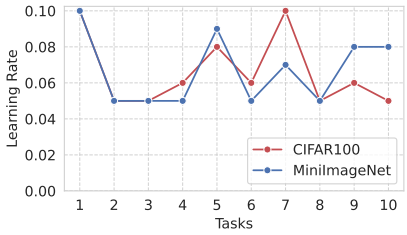
Search Space	[1,1000]	[1,25000]	[1,50000]	[1,10000]
EWC	9.17	22.02	**22.39**	21.09
Search Space	[1,10]	[1,25]	[1,50]	[1,100]
LwF	23.99	25.49	26.73	**28.9**

A.2 Hyperparameter Dynamics

In this section, we present an overview of the selected hyperparameters for the EWC and LWF methods. Our analysis highlights that our adaptive approach allows all models to flexibly adjust their hyperparameters across different tasks as illustrated in Figure 6 and 7. This flexibility plays a role in improving the performance of the models over time by adaptively adjusting hyperparameters.

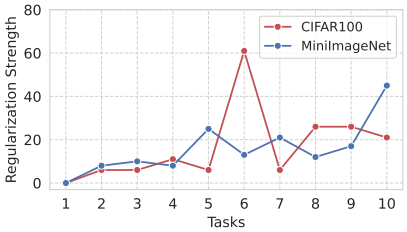


(a) Selected regularization strength

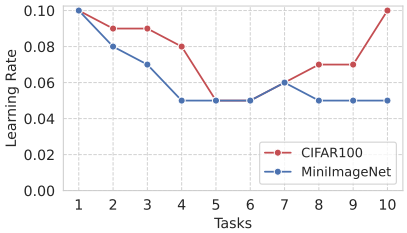


(b) Selected learning rate

Figure 6: Adaptive adjustments in (a) regularization strength and (b) learning rate for EWC across various task sequences and datasets.



(a) Selected regularization strength



(b) Selected learning rate

Figure 7: LwF's (a) regularization strength and (b) learning rate change dynamically across different task sequences and datasets.