INCREMENTAL LEARNING WITH TASK-SPECIFIC ADAPTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Incremental learning aims to continuously acquire new knowledge while retaining previously learned information. Existing literature primarily focuses on improving model, often stability at the cost of plasticity, to prevent the forgetting of earlier tasks. In this paper, we argue that inter-task differences are the primary cause of catastrophic forgetting. To address this, we propose a network architecture consisting of two distinct blocks: one for learning invariant features shared across tasks and another for capturing task-specific details. Specifically, we repurpose adapters, originally introduced for parameter-efficient fine-tuning, as feature modifiers to capture task-specific details, while the backbone network learns invariant features. Our approach integrates seamlessly with established methods, such as Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF). Extensive experiments on CIFAR-100 and ImageNet datasets demonstrate that our adapter-based methods consistently outperform non-adapter counterparts across diverse learning scenarios, including various task orders and data scales. Our approach improves both plasticity and stability, effectively addressing the stability-plasticity dilemma.

1 INTRODUCTION

Real-world data often arrive in a sequential manner, in batches, or through periodic updates. These data dynamics are further complicated by constraints such as limited storage capacity and privacy considerations. In such contexts, employing concurrent or multitask learning, where models are trained on a single, static, large dataset, can be costly or impractical. Incremental learning, also known as lifelong learning, is specifically designed to handle these dynamic environments. This paradigm allows a model to continuously learn and update its knowledge from new data without retraining from scratch. Unlike traditional methods that require access to the entire dataset for training, incremental learning enables the model to adapt to new tasks over time using only the data provided for each new task while retaining knowledge from previous tasks.

Feature extraction and fine-tuning are two common approaches for learning new tasks without accessing previous training data. In feature extraction, the weights of a pre-trained model are kept fixed, and the outputs of the top layer are used as features for the new task (Donahue et al., 2014; Belouadah & Popescu, 2018). While this approach helps maintain existing knowledge, it may struggle to effectively learn new tasks. Fine-tuning improves upon feature extraction by updating the model weights for the new task. A low learning rate is typically used to ensure the model retains the structure and knowledge from previous tasks (Girshick et al., 2014). While this approach enables better adaptation to new tasks, it carries the risk of *catastrophic forgetting*, where performance on previously learned tasks deteriorates rapidly (Goodfellow et al., 2013; McCloskey & Cohen, 1989). Forgetting occurs when the task-specific weights of the previous tasks are altered to accommodate new tasks. These methods may not fully address the challenges of incremental learning, which requires balancing the preservation of existing knowledge with the acquisition of new skills.

An ideal incremental learning algorithm should at least possess the following key properties: efficient memory usage and stability-plasticity balance. First, the algorithm should optimize memory usage, preserving no or only essential data points from previous tasks. This may reflect realistic scenarios where data might be transient or unavailable due to privacy or memory constraints. Second, it should maintain a balance between stability (retaining past knowledge) and plasticity (learning

new information), avoiding catastrophic forgetting (losing old knowledge when learning new data). Balancing stability and plasticity is a fundamental challenge in incremental learning, often referred to as the stability-plasticity dilemma (Mermillod et al., 2013).

Existing research in incremental learning primarily focuses on mitigating catastrophic forgetting by improving stability at the expense of plasticity. One popular approach is to directly regularize weight changes (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018) or to limit divergence in output predictions, between the old and new models (Li & Hoiem, 2017; Dhar et al., 2019). This strategy can be extended by incorporating data retained from previous tasks, which further prevents forgetting by replaying some of the prior knowledge (Chaudhry et al., 2018; Rebuffi et al., 2017; Zhang et al., 2020b). However, enhancing stability often hurts plasticity. Few works simultaneously improve stability and plasticity.

Contribution In this paper, we argue that the inter-task differences contribute to catastrophic forgetting and propose to model these differences. Specifically, we propose a network design consisting of two blocks: a backbone network for learning invariant features across all tasks and many small networks for modeling task-specific knowledge. Specifically, we use adapters, originally introduced for parameter-efficient fine-tuning of large language models (Houlsby et al., 2019), as these small networks to capture the task-specific information. Our approach differs from the conventional use of adapters, which are typically added to a frozen network to attain comparable performance to full fine-tuning. Instead, we re-purpose adapters as feature modifiers and train them together. This strategy enables the adapters to encapsulate task-specific information in the layers closer to the output, while squeezing task-invariant knowledge into layers nearer the input. Our approach can be integrated with many existing methods such as EWC and LwF. Our empirical results demonstrate that various adapter-assisted methods consistently outperform non-adapter counterparts along the learning process, ranging from regularization-based to rehearsal-based approaches, across both CIFAR-100 and ImageNet datasets. The advantage remains robust among dataset choices, task scales, and task orderings. Our approach improves both plasticity and stability, eliminating the stability-plasticity dilemma.

2 Related work

This section reviews works that are closely related to ours.

Multi-task Learning Multi-task learning involves training on all tasks simultaneously, which enhances performance through shared network parameters that exploit inter-task correlations, in contrast to incremental learning (Caruana, 1997; Ruder, 2017). However, multi-task learning can be costly or impractical due to the substantial computational burden of training on large volumes of data, as well as challenges related to limited storage capacity and privacy considerations (Kendall et al., 2018).This makes incremental learning an essential area of focus.

Incremental Learning Incremental learning (IL), also known as lifelong learning, involves learning from a sequence of tasks incrementally. However, this approach is prone to catastrophic forgetting, where learning new tasks can overwrite prior knowledge (McCloskey & Cohen, 1989). To address catastrophic forgetting, three primary strategies are commonly employed: regularization-based, rehearsal-based, and parameter-isolation methods. Regularization-based methods mitigate forgetting by regularizing differences in weights or output predictions between the old and new models (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Li & Hoiem, 2017; Dhar et al., 2019; Joseph et al., 2022). Rehearsal-based methods reinforce the retention of prior knowledge by storing some instances from previous tasks and training on a combined dataset of these instances and the new task data. This subset may include exemplar images from old tasks (Rebuffi et al., 2017; Chaudhry et al., 2018), publicly available external data (Lee et al., 2019; Zhang et al., 2020b), or synthetic data generated by generative models (Shin et al., 2017; Kemker & Kanan, 2017; He et al., 2018). Parameter-isolation methods provide substantial stability by freezing parameters that significantly contribute to prior tasks (Mallya & Lazebnik, 2018; Serra et al., 2018). Each of these methods has its own set of limitations. Regularization-based and parameter-isolation approaches enhance model stability but often reduce plasticity. Meanwhile, rehearsal-based methods may be impractical due to constraints related to storage capacity, privacy concerns, and scalability issues.

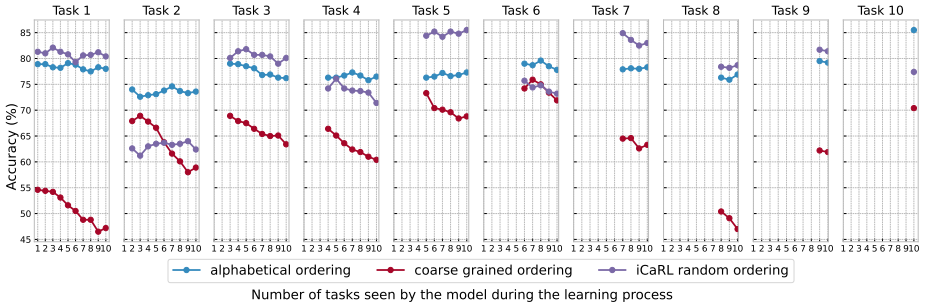

Figure 1: The model's accuracy using LwF for incremental learning is evaluated on three different CIFAR-100 task orderings: the standard alphabetical category ordering, the coarse grained ordering, and a random ordering with the fixed seed 1993 used by iCaRL (Rebuffi et al., 2017). The coarse grained ordering has more inter-task diversity.

Recent works have focused on task-specific modules or algorithms to mitigate catastrophic forgetting (Rajasegaran et al., 2020; Bhat et al., 2023; Pham et al., 2021; Wang et al., 2022; Liang & Li, 2024; Zhang et al., 2020a). Some of these approaches incorporate adapter-like subnets in various ways to ensure effective training. However, they are limited in several ways: 1) Some freeze the backbone adapter (Liang & Li, 2024; Zhang et al., 2020a), which may negatively impact performance; 2) Some use custom, complex loss functions (Liang & Li, 2024; Bhat et al., 2023) that have limited compatibility with new algorithms and restrict their broader applicability; 3) None explore regularization or prediction-based approaches. By focusing solely on network architecture, these methods overlook valuable insights from the dated but decent baselines such as LwF (Li & Hoiem, 2017). We aim to improve upon this by using adapters, providing a lightweight, compatible solution with strong performance derived from both adapter design and algorithmic insights.

Fine-tuning with Adaptors Another line of research explores the potential of adapters in transferring knowledge into downstream tasks, particularly within large language models. Adapters are small modules that are added between layers of a pre-trained large model (Houlsby et al., 2019). These modules are often tuned while keeping the original network frozen. Fine-tuning with adapters demonstrates comparable performance to full fine-tuning across diverse tasks, with only updating a small set of parameters efficiently (Li & Liang, 2021). Adapters also enable quick adaptation to new tasks without catastrophic forgetting (Pfeiffer et al., 2020a), addressing multi-domain (Chronopoulou et al., 2023; Asai et al., 2022) or multilingual problems (Pfeiffer et al., 2020b).

3 Incremental Learning with Task-Specific Adapters

This section identifies a key limitation of current knowledge distillation methods for incremental learning and introduces the adaptors as feature modifiers to address this issue. Additionally, we propose modifications to the existing methods in order to align with adapters and enhance the performance.

3.1 What is missing in regularization-based methods?

A key limitation of regularization-based methods is their susceptibility to the stability-plasticity dilemma. Specifically, anchoring the model to its performance on prior tasks limits its plasticity in learning new tasks, while relaxing the regularization leads to catastrophic forgetting. This issue becomes more pronounced with greater inter-task diversity in the incremental learning problem. Variations in inter-task diversity can be introduced through different task orderings.

Figure 1 examines the impact of task orderings on the classification performance of the LwF algorithm applied to CIFAR-100. In this scenario, the model is incrementally trained with 10 classes per task over a total of 10 tasks. Three different task orderings are evaluated: an alphabetical class ordering, a random ordering with a fixed seed commonly used by iCaRL and other methods (Rebuffi et al., 2017;

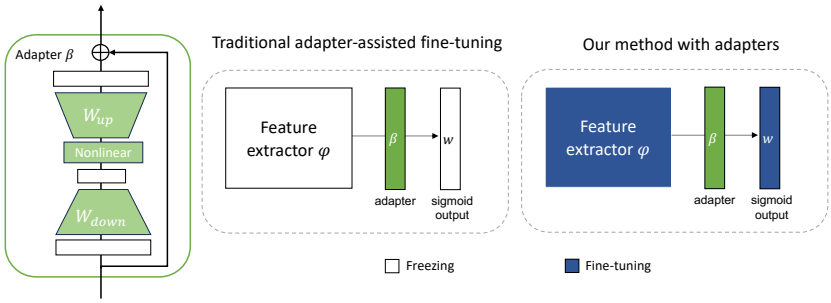

Figure 2: Architecture of the adapter and a comparison highlighting the distinctions in its implementation between traditional fine-tuning and our method. Left: an adapter consists of the down-projection, the nonlinear transformation, up-projection, and skip-connection. Right: The key difference between traditional use of adapter and ours is that we allow adapters to be co-trained with the entire network when learning a new task.

Zhang et al., 2020b), and a coarse grained ordering that groups similar classes within each task based on CIFAR-100's 20 coarse categories.

The coarse grained ordering has greater inter-task diversity, providing a way to assess how increased inter-task differences affect the incremental learning algorithm. Notably, when classes are learned in a coarse-grained ordering, there is a significant increase in both forgetting and accuracy loss for each task compared to other orderings. This increased forgetting and accuracy decline can be attributed to the stability-plasticity dilemma of the LwF algorithm when confronted with greater inter-task diversity.

Therefore, there is a need to develop new strategies that can eliminate the stability-plasticity dilemma of regularization-based approaches, enabling them to learn new tasks without forgetting.

3.2 Introducing adapters

In this section, we propose a network design consisting of two components: a backbone network for learning invariant features shared across all tasks, and multiple small networks for modeling task-specific information. The small networks enhance the plasticity of the architecture, enabling it to learn new tasks, while the backbone network maintains stability. Training over more tasks and samples allows the backbone network to be further refined, resulting in improved stability. This architectural design can enhance both stability and plasticity, effectively resolving the stability-plasticity dilemma.

Adapters are small, trainable modules initially proposed as an efficient way to fine-tune a pre-trained large language model for various text classification tasks (Houlsby et al., 2019). When using adapters to fine-tune a pre-trained network, the original network parameters remain fixed, making it a parameter-efficient fine-tuning strategy. Adapters demonstrate near state-of-the-art performance in natural language processing tasks compared to full fine-tuning.

In contrast to the efficient parameter fine-tuning literature where trainable adapters are added to a frozen pre-trained network, we use adapters as task-specific feature modifiers and co-train them along with the original network. These adapters are positioned between the backbone feature extractor φ and the label predictor layer, serving as task-specific feature modifiers β^t for each task t. The adapters, illustrated in Figure 2, adopt a conventional bottleneck structure. Starting with an initial dimension d and a bottleneck width b, we design the down-projection layer to reduce dimensionality from d to b using a fully-connected neural network with a weight matrix $\mathbf{W}_{d\times b}$ and a non-linear activation function g, expressed as $\texttt{Down}_{d\to b}(\mathbf{x}) = g\left(\mathbf{x}\mathbf{W}_{d\times b}\right)$. Similarly, the up-projection layer is defined as $\texttt{Up}_{b\to d}(\mathbf{x}) = g\left(\mathbf{x}\mathbf{W}_{b\times d}\right)$. Our adapters consist of both a down-projection and an up-projection step and are connected to the output via a skip-connection (He et al., 2021). This bottleneck design allows the adapter to utilize both backbone features from $\varphi(x)$ and the modified features processed through the down- and up-projection layers:

$$\beta^t(\varphi(x)) = \varphi(x) + \texttt{Up}^t_{b\to d}\left(\texttt{Down}^t_{d\to b}\left(\varphi(x)\right)\right).$$

Our adapter module incorporates three key features explicitly designed for incremental learning: (i) a compact design, characterized by a small number of parameters; (ii) the adaptation and enhancement to current constraints posed by existing methods; and (iii) the simultaneous updating of feature extractor layers and adapters. Specifically, the first addresses memory constraints, while the latter two contribute to both improved stability and improved plasticity.

For (i), our method utilizes adapters to control the growth rate of the overall model size when accommodating additional tasks. When applied to large networks, the parameters introduced by adapters are negligible, making our approach suitable for models with memory constraints. This approach contrasts with strategies that rely on dynamically expanding networks (Yoon et al., 2017; Yan et al., 2021).

For (ii) and (iii), we aim to address the stability-plasticity dilemma by the novel, two-block design and the co-training of them. Specifically, adapters are trained to improve the plasticity, and the continuous training of the backbone networks aims to enhance stability. Achieving such distinction is especially hard in the traditional, non-adapter networks since a holistic architecture inevitably leads to the above-mentioned dilemma.

To ensure that the adapters can align with the existing methods and effectively model inter-task differences, we employ method-specific regularization techniques. For the weight-regularized methods such as EWC, we free adapters from regularization; for the prediction-regularized methods such as LwF, we add a novel regularization to gain backbone-only distillation. The approaches and designs are introduced in Section 3.2.1.

3.2.1 Integrating Adapters into Existing Methods

We perform necessary modifications to the incremental learning methods so that the adapters can fit them and perform well. This section presents the adjustments for such integrations with different regularization methods.

Weight-regularized Methods The weight-regularized methods control and regularize the weights of each task. Taking Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), a noteworthy representative of such methods, as an example, $\mathcal{L}^t$, the loss at task t can be computed by the task loss and an additional term for regularizing each parameter:

$$\mathcal{L}^t = \ell^t(\theta) + \sum_{t'=1}^{t-1} \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta^*_{t',i})^2,$$

where θ is the model parameter and F_i is the Fisher information matrix at each parameter i.

The regularization itself enables the stability of the entire network. In order to improve the plasticity, we attempt to control the backbone's weight only so that the adapters remain unregularized and thus are able to move freely for parameter exploration. Hence, the integration of adapters to such methods is achieved by ruling out the adapter parameters from regularization, i.e. $i \notin \theta_a$:

$$\mathcal{L}^t = \ell^t(\theta) + \sum_{t'=1}^{t-1} \sum_{i \notin \theta_a} \frac{\lambda}{2} F_i (\theta_i - \theta^*_{t',i})^2.$$

This modification is applicable to all the weight-regularized methods, as long as they involve parameter-level consolidation. Methods such as Memory Aware Synapses (MAS) (Aljundi et al., 2018) and Path Integral (Path Integral) (Zenke et al., 2017) fall into this category and thus apply the above-mentioned adjustments to align with adapters.

Prediction-regularized Methods The prediction-regularized methods attempt to address the stability-plasticity dilemma through model distillation instead of weight control. Algorithms such as LwF and Learning without Memorizing (LWM) (Dhar et al., 2019) fall into this category. Additional to the task loss, the model outputs at task t are regularized with the model outputs at all tasks t' where

$1 \leq t' < t$, i.e. a distillation loss:

$$\begin{aligned}\mathcal{L}^t &= \ell^t(\theta) + \lambda_{\text{distill}} R^t_{\text{distill}} \\ &= \ell^t(\theta) + \lambda_{\text{distill}} \sum_{t'=1}^{t-1} M\left(\varphi^{t'}(x), \varphi^t(x)\right)\end{aligned}$$

where M is a metric that quantifies the similarities between the adapter outputs, such as the cosine similarity or cross entropy[1], and λ_{distill} is a hyperparameter to control the distillation term.

This regularization method loosens the stability restriction of the network by distilling instead of direct controlling model parameters. It allows the model to search for an optimal solution in a larger parameter space. However, while we anticipate the adapter and the backbone model to capture the task-specific and task-invariant information respectively, this distillation constraint neither gives the adapters absolute freedom to the parameter search space nor poses a strong restriction for backbone task-invariance. The former cannot be done since adapters are involved in the forward pass and the computation of the distillation loss, and is arguably not a big deal due to that the restriction is already-loosen compared to the weight-regularized methods. Addressing the latter problem, we introduce an additional backbone regularization:

$$R^t_\varphi = \sum_{t'=1}^{t-1} M\left(\text{Linear}_{d\times c}(\varphi^{t'}(x)), \text{Linear}_{d\times c}(\varphi^t(x))\right),$$

where $c \leq d$ is a dimension we reduce to. In practical, we choose c to be the number of classes of each task, as intuitively we make this regularization implicitly a direct distillation on backbones.

Hence, to learn a new task t, we define a loss function that includes the task loss, the distillation term, and the backbone regularizer to align backbones across tasks to improve stability:

$$\mathcal{L}_t = \ell_t(\theta) + \lambda_{\text{distill}} R^t_{\text{distill}} - \lambda_\varphi R^t_\varphi, \tag{1}$$

where λ_φ is the hyperparameter for the backbone regularizer. The parameter λ_{distill} aims to balance retaining prior knowledge with adapting to new tasks during the incremental learning process, while λ_φ controls the direction regularization to the backbone across tasks. We apply this adapter regularization on LwF as it is a prediction-regularized method and, in our experience, it is among the most effective methods for incremental learning tasks.

4 Experiments

In this section, we compare our framework on various methods to their counterparts that are not using adaptors. We investigate impact of adapters on different settings, including different method types, task scales, task orderings and datasets.

4.1 Experimental setup

Datasets We compare the peformance of different methods on two datasets: CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Russakovsky et al., 2015). CIFAR-100 consists of images with small resolutions (input sizes of $32 \times 32 \times 3$) and serves as our primary focus for studying the impact of adapters across different settings. We also include ImageNet, which offers more diverse training images at higher resolutions ($224 \times 224 \times 3$). To mitigate training time and resource constraints, we limit our analysis to the first 100 classes from ImageNet. Dataset statistics are summarized in Appendix A.1.

To ensure a fair comparison, we perform hyperparameter tuning and learning rate selection on a validation set (Masana et al., 2022). The validation set is a class-balanced split, compromising 10% samples from the original training dataset, while the remaining 90% serves as our training dataset. Details on the hyperparameter tuning and learning rate selection can be found in Appendix A.2, and the selection of adapter-specific hyperparameters is explained in Section 4.2.

[1]Following previous works, we use the cross entropy loss as the metric.

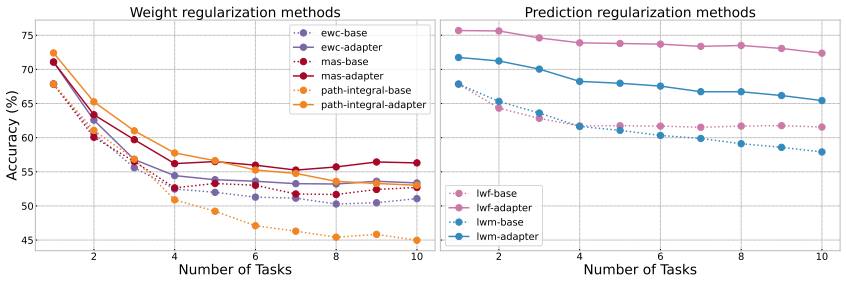

Figure 3: The average accuracy for regularization-based methods with or without adapters on CIFAR-100 (alphabetical ordering) in task-IL. The solid line represents the results with adapter, while the dashed line represents the results without adapter. The left figure displays the performance with weight regularization (EWC, MAS, and Path Integral), and the right figure displays the performance with prediction regularization (LwF and LwM).

Network architectures Following (De Lange et al., 2022; Hou et al., 2019), we employ two different models for the two datasets. For CIFAR-100, which contains small-resolution images, we use ResNet-34. For ImageNet, with larger-resolution images, we use ResNet-18, as suggested in (He et al., 2016).

Evaluation metrics Two evaluation protocols are commonly used in incremental learning: task incremental learning (task-IL) and class incremental learning (class-IL). Task-IL evaluates the network in a multi-head setting, utilizing a task-ID oracle to determine the appropriate task-specific head at the inference time. In task-IL, the model does not need to differentiate between classes from different tasks. In contrast, class-IL presents a more practical yet challenging scenario where the model must make predictions across all learned classes within a single-head configuration. This requires the model to resolve confusion arising from different tasks. In this section, we focus on task-IL with task-ID information at the inference time , while results for class-IL are included in Appendix B.

To compare the overall learning process across different methods, we use the average accuracy at each task t, denoted by $A_t = \frac{1}{t}\sum_{i=1}^{T} a_{t,i}$, and $a_{t,k}$ is the accuracy evaluated on task k after training on task t. To ensure reliable and consistent results, we report the averaged results over 10 runs with different random seeds for both CIFAR-100 and ImageNet.

4.2 Experimental results

On regularization-based methods In this section, we study the effect of adapters combined with various weight-regularized methods in incremental learning, including Elastic Weight Consolidatio EWC (Kirkpatrick et al., 2017), Memory Aware Synapses (MAS) (Aljundi et al., 2018), Path Integral (PathInt) (Zenke et al., 2017), as well as prediction-regularized methods such as Learning without Forgetting (LwF) (Li & Hoiem, 2017) and Learning without Memorizing (LwM) (Dhar et al., 2019).

Figure 3 compares different methods with and without adapters in task-IL on CIFAR-100. From the first task onward, weight-regularized methods with adapters exhibit an approximate 3% increase in average accuracy compared to those without adaptors. This improvement is consistently maintained throughout the learning process across all methods. For prediction-regularized methods, the accuracy advantage further escalates to as much as 5%. The observed increase in accuracy can be primarily attributed to the model's improved plasticity to learn new task-specific knowledge through the use of adapters, while the backbone was continuously trained as well.

On task scale This paragraph examines the effect of the task scale. The benefits of utilizing adapters diminish as the number of classes increases within each task. As illustrated in Figure 4, when learning either 5 or 10 classes simultaneously, our adapter-based approach continues to effectively capture inter-task differences and significantly outperforms methods without adapters. However, while the advantages are still present, they become less pronounced as the number of classes per task increases. This reduction in the performance margin is understandable, as learning more classes per task not only provides more data for the model to learn but also requires more memory and storage, with fewer

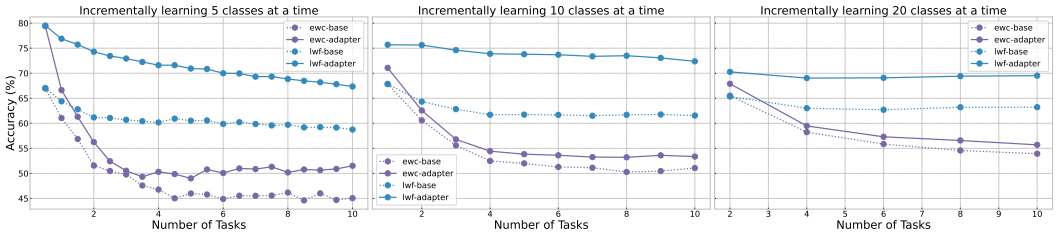

Figure 4: The average accuracy for EWC and LwF with learning 5, 10, and 20 classes at a time on CIFAR-100 (alphabetical ordering) in task-IL.

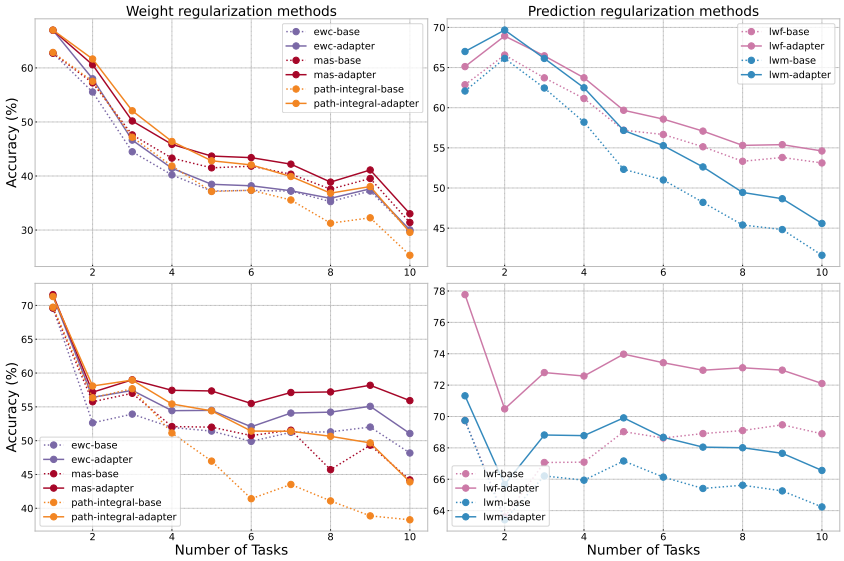

Figure 5: The average accuracy for regularization-based methods with or without adapters on different orderings of CIFAR-100 in task-IL. The upper two figures present the experimental results on the coarse ordering, and the lower two are on the iCaRL ordering. The solid line represents the results with adapter, while the dashed line represents the results without adapter.

regime shifts. In the extreme case, when the number of classes per task reaches 100, incremental learning reduces to a multi-task learning problem.

On task ordering As discussed in Section 3.1 and by Masana et al. (2020), the impact of task orderings on the performance of incremental learning models is often overlooked. While regularization-based methods suffer from the stability-plasticity dilemma, the introduction of adapters improves both stability and plasticity. This resolution of the stability-plasticity dilemma is attributed to the fact that task-specific knowledge is effectively captured by the adapters, while the backbone continuously learns invariant knowledge. Since all prior experiments discussed in this section were conducted using the CIFAR-100 alphabetical ordering, we have further evaluated all methods using additional orderings, specifically the coarse-grained ordering and the iCaRL ordering.

As shown in Figure 5, methods with adapters are indeed influenced by the varying difficulties associated with different orderings. Although the advantage persists, it diminishes to approximately 1% in some cases. Nonetheless, a general superiority remains evident, as most methods maintain a 3% margin over their non-adapter counterparts across all orderings. In every ordering scenario, adapters consistently exhibit the best overall performance in incremental learning.

On Imagenet This section evaluates our method's performance across larger domain shifts by assessing its performance on ImageNet-Subset, a significantly larger dataset compared to CIFAR-100.

Method	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
MAS	80.4	73.6	74.4	71.3	72.8	72.9	73.5	72.1	72.7
EWC	80.3	74.6	72.0	67.8	63.2	63.9	63.6	61.4	60.8
PathInt	53.9	38.5	33.7	30.4	28.7	29.0	28.9	28.2	27.1
LwF	82.6	77.7	76.8	75.2	73.9	73.7	72.3	70.0	68.2
LwM	81.8	76.3	74.3	70.9	68.4	66.2	64.0	60.3	58.0
MAS-A	80.0	73.6	74.0	72.2	73.3	**74.6**	**75.0**	**74.2**	**74.2**
EWC-A	76.0	67.7	68.0	67.3	67.2	68.3	67.3	65.7	65.3
PathInt-A	76.9	68.3	67.3	65.4	65.5	67.1	67.1	65.0	65.0
LwF-A	**83.8**	**79.8**	**78.3**	**76.2**	**74.2**	73.0	71.6	69.0	67.2
LwM-A	82.8	75.9	73.9	70.6	67.8	65.9	63.2	59.4	56.9

Table 1: The average accuracy for regularization-based methods with or without adapters on ImageNet subset in task-IL. Methods without the "-A" suffix represent the baseline, while those with the suffix include adapters. Following experiments conducted on CIFAR-100 in Section 4.2, these adapters are configured with bottleneck width 128.

Our method faces certain limitations when applied to ImageNet, as selecting adapter hyperparameters becomes prohibitively expensive on such a large dataset with an average of 10 seeds, which led us to apply the CIFAR-100 hyperparameter setting directly to ImageNet. Additionally, the use of task-specific adapters slows down the generalization process compared to their non-adaptor counterparts. This slowdown occurs because non-adapter methods do not distinguish between task-sharing and task-specific patterns and thus are less impacted on ImageNet, which we only run for 50 epochs as mentioned in Appendix A.2.

While we recommend a more comprehensive study involving careful hyperparameter selection and extended training epochs, our current experimental results, as shown in Table 1, indicate that methods with adapters yield the best performance across all incremental tasks. Even with the hyperparameters from Section 4.2, which are tuned using CIFAR-100 and may not be optimal, methods with adaptors still demonstrate non-trivial performance imoprovement.

On modern IL methods There is a growing body of incremental learning methods that incorporate task-specific components. We conducted experiments to address two key questions: 1) Can adapters be integrated with these methods, and does such integration further improve performance? 2) Does our adapter integration and training paradigm outperform existing adapter-based methods? For the first question, we integrate adapters with DualPrompt (Wang et al., 2022) and iTAML (Rajasegaran et al., 2020). For the second, we directly compare our approach with TAMiL (Bhat et al., 2023) by aligning our setup with theirs. The experiment details can be found in Appendix A. The results, shown in Table 2, indicate that integrating adapters boosts the original frameworks' performance by more than 1%, and our method outperforms TAMiL, a counterpart that uses attention modules.

4.3 Ablation Studies

The bottleneck width choice The choice of adapter bottleneck width is crucial to the model performance. We selected EWC (Kirkpatrick et al., 2017) and LwF (Li & Hoiem, 2017) as baseline methods, due to their strong performance among weight-regularized and prediction-regularized methods, respectively. As illustrated in Figure 6, adapters with a bottleneck width of 256 consistently ranked among the top configurations.

Training with backbone frozen Various works propose adapter-like network architectures where the backbone is frozen (Liang & Li, 2024; Zhang et al., 2020a). While our framework adapts the backbone to capture task-invariant information and the adapters to capture task-specific information, we hypothesize that freezing the backbone does not support incremental learning. This is because the backbone still requires updates with new, task-inspecific knowledge, which may conflict with the prior knowledge acquired during pre-training. To investigate this, we conduct experiments using the LwF method, where both models are trained with adapters, but one freezes the backbone. Our results, shown in Table 2, demonstrate that the co-trained model, which does not freeze the backbone,

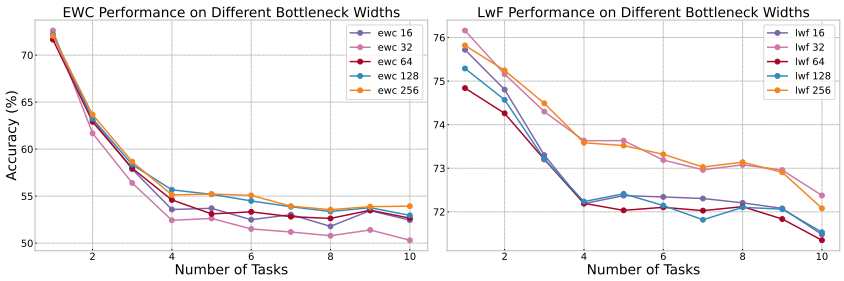

Figure 6: The performance of EWC and LwF methods with different adapter bottleneck width choice on the CIFAR-100 dataset (alphabetical ordering) in task-IL. The suffix `16/32/64/128/256` indicates the method implemented with width 16, 32, 64, 128, and 256, respectively.

outperforms the counterpart. This supports our hypothesis regarding the impact of freezing the backbone.

Methods	Acc.	Methods	Acc.	Methods	Acc.	Methods	Acc.
DualNet	88.2	iTAML	79.0	TAMiL	71.4	LwF-A	**74.0**
DualNet-A	**89.3**	iTAML-A	**80.1**	Adapter+LwF	**74.7**	Lwf-A-FrB	72.9

Table 2: From left to right: DualNet vs. DualNet+adapter, iTAML vs. iTAML+adapter, TAMiL vs. Adapter+LwF (The best method-adapter pair we yielded), and Lwf-A (co-trained) vs. LwF-A (**Fr**ozen **B**ackbone). Test conducted on CIFAR-100, task-IL, top-1 accuracy averaged with 10 tasks is reported.

5 Conclusion

In this paper, we propose a network design consisting of two blocks: a backbone network for learning invariant features across all tasks and multiple adapters for modeling task-specific knowledge. The backbone and adapters are co-trained continuously in an incremental learning framework. Our extensive experiments conducted on CIFAR-100 and ImageNet, across various orderings and task scales, show that introducing task-specific adaptors consistently improves the performance of all considered methods, and exhibits extensive compatibility with all of them. Consequently, we offer an effective solution to resolve the stability-plasticity dilemma for incremental learning, and we envision that future IL algorithms can be benefited from our work, a simple but effective integration of adatpers.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 139–154, 2018.

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6655–6672, 2022.

Eden Belouadah and Adrian Popescu. Deesil: Deep-shallow incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 0–0, 2018.

Prashant Bhat, Bahram Zonooz, and Elahe Arani. Task-aware information routing from common representation space in lifelong learning, 2023. URL `https://arxiv.org/abs/2302.11346`.

Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*, 2023.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8250–8259, 2021.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2022.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5138–5146, 2019.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pp. 647–655. PMLR, 2014.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *BMVC*, pp. 98, 2018.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

KJ Joseph, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Vineeth N Balasubramanian. Energy-based latent aligner for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7452–7461, 2022.

Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

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

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

Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 312–321, 2019.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

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

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning, 2024. URL https://arxiv.org/abs/2404.00228.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 2018.

Marc Masana, Bartłomiej Twardowski, and Joost van de Weijer. On class orderings for incremental learning, 2020. URL https://arxiv.org/abs/2007.02145.

Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020a.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Unks everywhere: Adapting multilingual language models to new scripts. *arXiv preprint arXiv:2012.15562*, 2020b.

Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow, 2021. URL https://arxiv.org/abs/2110.00175.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach, 2020. URL https://arxiv.org/abs/2003.11652.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning, 2022. URL https://arxiv.org/abs/2112.08654.

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

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 3987–3995, 2017.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: A baseline for network adaptation via additive side networks, 2020a. URL https://arxiv.org/abs/1912.13503.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1131–1140, 2020b.

Appendix

Table of Contents

A Implementation details **14**
A.1 Datasets 14
A.2 Learning rate search and hyperparameter tuning 14
A.3 iTAML, TAMiL, and DualPrompt 15

B Class-IL results **15**

A Implementation details

We follow the framework FACIL (Masana et al., 2022), and code is implemented using Pytorch. We apply SGD with momentum set to 0.9 and weight decay set to 0.0002. Gradient clipping is set to 1.0.

A.1 Datasets

Table 3 presents a summary of the datasets used in the experiments. We apply data augmentation to both datasets, including padding, cropping, input normalization, and random horizontal flipping. Specifically, for CIFAR-100, we apply a padding of 4 pixels to each side of the image, followed by random cropping to 32×32 for training purposes and center cropping for testing. For the ImageNet-Subset, we resize the images to 256×256, then perform random cropping to 224×224 for training and center cropping for testing.

A.2 Learning rate search and hyperparameter tuning

We apply the Continual hyperparameter Framework (De Lange & Tuytelaars, 2021), a common framework to select learning rates (LRs) and tune hyperparameters for methods of incremental learning. The framework incorporates two phase: the Maximal Plasticity Search phase to search LR with fine-tuning on the new task, and the Stability Decay phase to search for the optimal hyperparameters.

For the Maximal Plasticity Search phase, we fine-tune the model on the new task and select the optimal LR to achieve high plasticity. Specifically, we train the model from scratch for the first task and apply LR search on $\{$5e-1, 1e-1, 5e-2$\}$. Starting from the second task, the LR search space is limited to 1e-1, 5e-2, 1e-2, 5e-3, 1e-3$\}$. LR decay is applied, with a decay factor of 3 and patience of 10 epochs. The stopping criteria is either the LR below 1e-4 or 100 epochs have passed.

For the Stability Decay phase, we fix the LR and select the hyperparameter starting from a high value, with a gradual decay to achieve the optimal stability-plasticity trade-off. We start from a high value of hyperparameter because this is close to freezing the network such that old knowledge is preserved, and through gradual decay the model becomes less intransigence and slowly converges towards higher forgetting, which would ultimately corresponds to fine-tuning of the previous phase. Specifically, we reduce the hyperparameter value in half if the method accuracy is below the 95% of the fine-tuning accuracy in the previous LR search phase.

Next, we present the starting values for hyperparameters across methods. Most of the methods discussed in our paper are regularization-based approach, which include a hyperparameter λ for the regularizer that controls the trade-off between stability and plasticity. The starting values for λ are summarized in Table 5. For other method-specific parameters, we generally follow the corresponding original work. For instance, we fix the temperature parameter as 2 for LwF. For PathInt, the dampling parameter is set to 0.1. For LwM, the attention distillation parameter is set to 1 based on an empirical evaluation in (Masana et al., 2022).

Datasets	# Train	# Validation	# Test	Input size	Batch size
CIFAR-100	45,000	5,000	10,000	$32 \times 32 \times 3$	128
ImageNet-Subset	117,000	13,000	5,000	$224 \times 224 \times 3$	4

Table 3: Summary of datasets used. Both datasets contain 100 classes each.

Table 4: The average accuracy for regularization-based methods with or without adapters on ImageNet subset in class-IL. Methods without the "-A" suffix represent the baseline, while those with the suffix include adapters. These adapters are configured with the same bottleneck width and number of layers as those used in the CIFAR-100 experiments.

Method	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
MAS	68.5	52.7	47.9	40.4	**39.2**	**35.7**	**34.4**	31.3	30.0
EWC	68.2	53.5	45.0	36.0	29.1	26.4	24.2	21.6	19.7
PathInt	41.5	22.7	15.9	11.8	9.5	8.0	7.2	6.3	5.3
LwF	69.7	57.1	51.0	44.3	38.6	33.1	29.7	26.4	24.4
LwM	66.1	50.9	44.7	36.2	29.3	23.3	20.2	17.3	15.0
MAS-A	66.4	51.8	46.6	40.4	38.0	35.0	34.2	**31.9**	**30.1**
EWC-A	62.1	45.1	39.0	33.9	30.7	28.0	26.4	23.6	22.0
PathInt-A	63.0	44.5	36.1	31.1	28.7	26.4	25.1	22.7	20.7
LwF-A	71.5	**59.7**	**53.4**	**45.2**	38.4	32.1	28.5	24.6	22.0
LwM-A	**68.6**	50.7	43.0	34.2	26.9	22.1	19.3	16.3	13.9

hyperparameter	EWC	MAS	PathInt	LwF	LwF-reg	LwM
λ	10,000	400	10	10	5+0.5	2

Table 5: Summary of hyperparameter to control the stability and plasticity used across regularization-based methods. For the LwF with adapter and regularizations (LwF-reg), the two λ's ($\lambda_{distill}, \lambda_{\varphi}$) are set to 5 and 0.5 respectively.

Due to computational cost constraints, we have adjusted the stopping criteria for the ImageNet Subset to 50 epochs, rather than the previously mentioned 100. Additionally, we do not conduct LR searches or tune hyperparameters for this dataset, and fix the LR at 0.01.

A.3 iTAML, TAMiL, and DualPrompt

We implementd iTAML, TAMiL and DualPrompt with their default set of hyperparameters. Specifically,

- For iTAML, we run a ResNet 18 by 70 epochs with memory size 3000, $\mu = 1, \beta = 2$, and $r = 5$. Learning rate is set to 0.01.
- For DualPrompt, we run a ViT-base-patch16-224 with G-prompt and E-prompt enabled. The model is optimized with an adam optimizer with $lr = 0.03$. 2 epochs are run for each task.
- For TAMiL, We run a ResNet 18 by 50 epochs. The learning rate is set to 0.03, and exemplar size is set to 200. The seed is set to 10.

B Class-IL results

Figure 7 illustrates the average accuracy of all methods in the setup described in Section **??**, specifically for class incremental learning (Class-IL). Figure 8 demonstrates the results across various task scales, while Figure 9 presents the results based on different orderings. Table 4 provides the results for ImageNet.

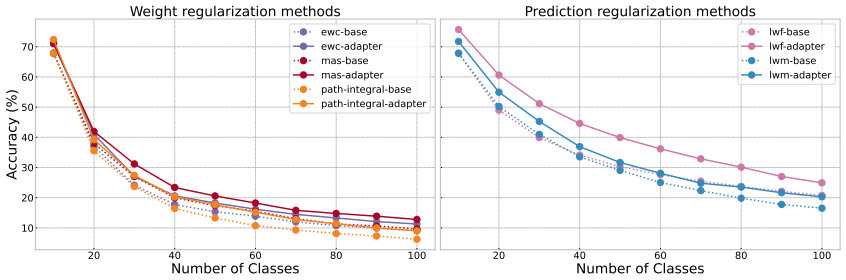

Figure 7: The average accuracy for regularization-based methods with or without adapters on CIFAR-100 (alphabetical ordering) in class-IL. The solid line represents the results with adapter, while the dashed line represents the results without adapter. The left figure displays the performance with weight regularization, and the right figure displays the performance with prediction regularization.

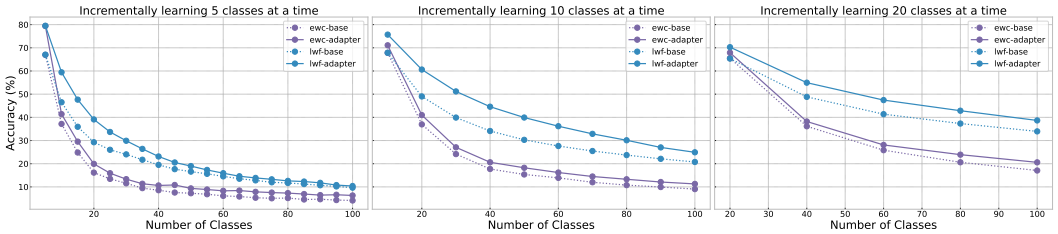

Figure 8: The average accuracy for EWC and LwF with learning 5, 10, and 20 classes at a time on CIFAR-100 (alphabetical ordering) in class-IL.

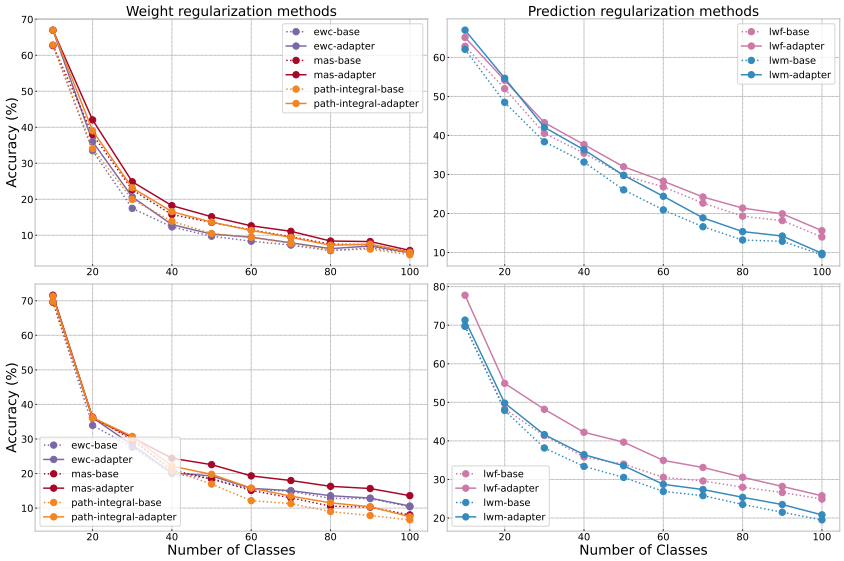

Figure 9: The average accuracy for regularization-based methods with or without adapters on CIFAR-100 (coarse ordering and iCaRL ordering) in class-IL. The solid line represents the results with adapter, while the dashed line represents the results without adapter.