Online-LoRA: Task-free Online Continual Learning via Low Rank Adaptation Supplementary Material

Xiwen Wei
The University of Texas at Austin
xiwenwei@utexas.edu

Guihong Li
The University of Texas at Austin
liguihong1995@gmail.com

Radu Marculescu
The University of Texas at Austin
radum@utexas.edu

In this supplementary materials, a unique labeling with an "S" prefix (e.g., S1, S2, etc.) is used, distinguishing them from the main paper references.

A. Evaluation Metrics

In this section, we present the definitions of the three evaluation metrics we used in our experiments, supplementing Section 4.2 in the main paper.

Let $a_{i,j}$ be the testing accuracy on the i^{th} task after training on j^{th} task. The total number of tasks is denoted by T.

Final Accuracy The final accuracy A_{Final} is calculated as the average accuracy across all tasks after training on the final task:

$$A_{\text{Final}} = \frac{1}{T}\sum_{i=1}^{T} a_{i,T} \tag{1}$$

Area Under the Curve of Accuracy The A_{AUC} (Area Under the Curve of Accuracy) is defined as the area under the curve (AUC) of the accuracy-to-# of samples curve [125]. To construct the curve, the accuracy is measure after each sample is observed. A_{AUC} measures the any time inference accuracy of the model:

$$A_{\text{AUC}} = \sum_{i=1}^{k} f(i \cdot \Delta n) \cdot \Delta n, \tag{2}$$

where the step size Δn is defined as $\Delta n = 1$, representing the number of samples observed between inference queries, and $f(\cdot)$ denotes the curve in the accuracy-to-{number of samples} plot. A high A_{AUC} indicates that the method consistently maintains high accuracy throughout training.

Forgetting Forgetting is defined as the averaged differences between the historical maximum accuracy of task k and the accuracy of task k after all tasks finish training:

$$\text{Forgetting} = \frac{1}{T-1}\sum_{k=1}^{T-1} \max_{t=1,2,\ldots,T-1}(a_{k,t} - a_{k,T}) \tag{3}$$

The last task T is excluded because the forgetting of the last task is always 0.

B. Experimental Details

In this section, we provide details of the experiments we reported in the paper, supplementing Section 4 in the main paper.

Data preprocessing Because we focus on the ViT architectures ViT-B/16 and ViT-S/16, all input images are resized to 224×224 and normalized to [0, 1].

Hyperparameters For tuning the threshold values for each dataset (CIFAR-100 [112], ImageNet-R [110], ImageNet-S [123], CUB-200 [122], and CORe50 [116]), we conducted a grid search following the protocol in [117]. The threshold grid is shown in Table S1. Table S2 shows the threshold values we used in our experiments. For CIFAR-100, ImageNet-R, and ImageNet-S, these threshold values remain consistent in both disjoint and Si-blurry class-incremental scenarios.

We set the regularization factor λ=2000.0 (see Equation 7 in the main paper) for all experiments.

C. Loss Surface

Figure S1 shows more qualitative examples of how the loss surface recognizes data distribution shifts, supplementing Section 3.2 in the main paper. MAS [103] introduces the *loss surface* to derive information about incoming streaming data in the task-free scenario. As shown in Figure S1, the peaks on the loss surface indicate shifts in the input data distribution. And the stable regions, namely plateaus, signal the convergence of the model. For instance, the Split

Threshold	CIFAR-100	ImageNet-R	ImageNet-S	CUB-200	CORe50
Mean	[2.2, 2.6, 2.8, 3.0]	[5.2, 5.4, 5.6, 5.8, 6.0]			[18.0, 24.0, 30.0]
Variance	[0.02, 0.03, 0.04, 0.06, 0.08, 0.1]				[0.6, 0.8, 1.0, 1.2]

Table S1. Hyperparameter grid for the mean and variance threshold values of the loss window in our Online-LoRA.

Threshold	CIFAR-100	ImageNet-R	ImageNet-S	CORe50	CUB-200
Mean	2.6	5.2	5.6	6.0	24.0
Variance	0.03	0.02	0.06	0.1	1.0

Table S2. Mean and variance thresholds of the loss window for different datasets.

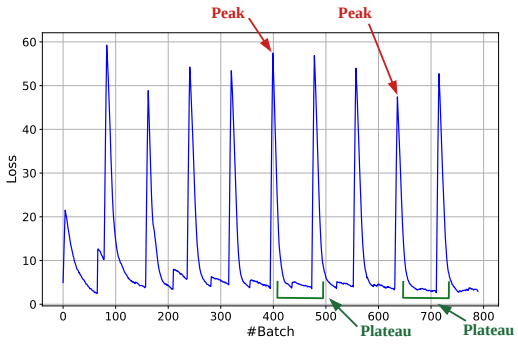


Figure S1. Loss surface of Online-LoRA on Split CIFAR-100 using ViT-B/16 model. Note that other peaks and plateaus exist but are not marked.

CIFAR-100 dataset has 10 distinct tasks, with the data distribution remaining constant within each task. As a result, during the learning process of Split CIFAR-100, there are 9 shifts in data distribution, corresponding to 9 peaks in the loss surface, as illustrated in Figure S1.

To identify plateaus on the loss surface, we employ a *loss window*, which is a sliding window that moves across consecutive training losses. Within this window, we closely observe both the mean and variance of the losses. A plateau is identified when both metrics fall below a predefined threshold (see Appendix B for details). Upon detecting a plateau, we proceed to introduce new LoRA parameters and update the estimation of the model parameter importance. Our goal in identifying plateaus is to mark periods of stable prediction following shifts in data distribution. Therefore, we only classify a phase as a plateau if it follows a peak. A peak is recognized when the loss window's mean increases by an amount exceeding the standard deviation of the window within a single batch.

D. Results of Swin Transformer

In this section, we present the results for the disjoint class-incremental and domain-incremental settings (for details on these settings, see Section 4.1 in the main paper) using the Swin Transformer architecture [115]. For a fair comparison, the hyperparameters for the baseline methods are set according to the descriptions in Appendix F.3. For our method, we use a learning rate of 0.0003 for the Swin Transformer.

As shown in Table S3, our approach consistently outperforms other baseline methods in both disjoint class-incremental and domain-incremental learning settings. This demonstrates that our method remains effective across various ViT architectures, extending beyond the ViT-B/16 and ViT-S/16 models reported in Section 4.3 of the main paper.

E. Supplementary Ablation Study

E.1. Ablation Study on Imagenet-S Dataset

In addition to the ablation results on Split Imagenet-R presented in Section 4.5 of the main paper, this section provides further ablation results on the Split Imagenet-Sketch dataset with varying task lengths. As shown in Table S4, our Online-LoRA consistently outperforms other variants that lack certain components. These results demonstrate that both the hard buffer loss and incremental LoRA, along with online parameter regularization, are crucial for the performance of our approach.

The baseline involves continuous fine-tuning of a single set of LoRA parameters. In contrast, Online-LoRA introduces an incremental LoRA architecture coupled with parameter importance-based regularization, and preserves a hard buffer along with its loss computations. Individually, each component improves performance and reduces forgetting. However, integrating both components into the baseline achieves the optimal performance, demonstrating the efficacy of our complete approach.

E.2. Impact of Pre-trained Weights

In this section, we demonstrate that our experimental settings do not provide any unfair advantage to our Online-LoRA approach through the use of pre-trained ViT models.

First, it is important to note that all baseline methods in our experiments utilize the same pre-trained ViT models as their backbones, just like Online-LoRA. Consequently, all methods benefit from the pre-training to varying extents, particularly those originally implemented with ResNet18 backbones (Table S6). For detailed information

Method	Split-ImageNet-S		CORe50	
	A_{Final} (↑)	*Forgetting* (↓)	A_{Final} (↑)	*Forgetting* (↓)
AGEM [106]	31.67±0.96	50.12±0.27	90.15±1.31	1.16±0.05
ER [107]	42.60±0.75	38.68±0.26	88.93±2.99	4.16±0.09
EWC++ [105]	29.57±1.57	51.87±0.04	90.91±1.28	0.04±0.02
MIR [102]	42.90±0.19	38.49±0.15	87.47±0.65	5.67±0.14
GDumb [120]	14.76±1.13	-	79.52±3.00	-
Ours	**53.75**±0.29	**32.86**±0.89	**95.29**±0.06	**0.00**±0.00
UB	71.98±0.23	-	97.56±0.02	-

Table S3. Results of disjoint class-incremental learning and domain-incremental learning using Swin Transformer. '↑' means higher is better and '↓' means lower is better. The best results are noted by **bold**. *UB* is the upper-bound performance. With Swin Transformer, our Online-LoRA method consistently outperforms other baseline methods across various settings, demonstrating its adaptability and effectiveness across different ViT architectures.

Incremental LoRA	Hard loss	10 tasks		20 tasks	
		A_{Final} (↑)	*Forgetting* (↓)	A_{Final} (↑)	*Forgetting* (↓)
-	-	30.66±0.25	38.70±0.40	24.49±2.61	39.29±2.57
✓	-	31.11±2.60	34.62±2.98	32.47±0.29	33.14±1.39
-	✓	36.26±0.12	39.29±2.57	35.43±4.99	32.56±2.72
✓	✓	**47.06**±0.24	**28.09**±3.25	**44.19**±2.09	**28.48**±0.24

Table S4. Ablation results of ViT-B/16 model on Split ImageNet-Sketch dataset. '↑' means higher is better and '↓' means lower is better. "Incremental LoRA": introducing new, trainable LoRA at each loss plateau with the model parameter regularization in Equation 7 in paper. "Hard loss": including $\mathcal{L}(F(X_B;\theta), Y_B)$ (the loss from hard buffer samples) in the final learning objective in Equation 6 in paper. A check mark (✓) indicates the presence of the component, while a dash (—) indicates its absence.

on the backbones used by each baseline, please refer to Appendix F.2.

Second, we show that simply using pre-trained models without applying any CL methods or strategies fails to yield competitive performance. To illustrate this, we introduce three simple baselines:

- **Frozen FT**: This baseline freezes the pre-trained backbone (feature extractor). Only the classification head (the final layer used for classification) is continuously fine-tuned on the data stream. Given that the model is pre-trained on the ImageNet-21K dataset, if any unfair advantage exists due to data leakage or other factors, it should be evident here by showing strong performance.

- **Continual FT**: This baseline fully fine-tunes the pre-trained model, including both the backbone and the classification head, on each new data batch. This is consistent with our OCL setting where the model encounters each data batch only once. If the pre-trained weights alone brings any unfair advantage, this baseline should perform competitively, similar to methods specifically designed for CL.

- **Random Head**: This baseline uses the pre-trained model's backbone with a newly initialized classifier head and performs only inference without any fine-tuning. Since the classifier head is randomly initialized, it should provide a clear lower bound for performance, demonstrating that without any adaptation or learning, the model's performance is essentially at chance level.

As shown in Table S5, **Random Head** baseline achieves near-zero accuracy, confirming that merely using pre-trained weights without adaptation to the test dataset does not have an advantage. Although the **Frozen FT** and **Continual FT** baselines outperform some CL methods (which also use the same pre-trained models), they still suffer from severe forgetting and exhibit a significant performance gap compared to other methods, particularly our Online-LoRA, with nearly a 20% difference in final accuracy and a 30% difference in forgetting.

These results demonstrate that the performance advantages of our Online-LoRA method over the baseline CL methods are not simply due to the use of pre-trained models. Instead, they arise from the effectiveness of our approach. The pre-trained weights provide a common foundation for

Method	Accuracy (↑)	Forgetting (↓)
Random Head	0.08±0.00	-
Frozen FT	27.98±0.29	55.12±0.43
Continual FT	28.49±0.21	53.49±0.07
AGEM [106]	5.60±2.74	53.97±1.97
ER [107]	40.99±3.96	32.38±0.89
EWC++ [105]	3.86±2.02	56.95±1.46
MIR [102]	41.51±2.99	31.32±5.17
GDumb [120]	1.65±0.22	-
PCR [114]	46.11±3.03	25.50±0.41
DER++ [104]	30.90±8.04	24.26±4.14
LODE (DER++) [113]	42.20±6.46	31.83±1.05
EMA (DER++) [121]	41.75±1.98	32.65±1.55
EMA (RAR) [121]	30.04±0.33	39.36±0.04
Online-LoRA (ours)	**48.18±0.63**	**23.85±1.48**
UB	63.82±0.02	-

Table S5. Performance comparison between pre-trained models without CL strategies and pre-trained models with CL strategies on Split ImageNet-R (online class-incremental learning setting). ViT-B/16 backbone is used. While some methods do not outperform simple fine-tuning on a continuous data stream, other CL methods provide significant performance improvements to the pre-trained model. This demonstrates that the advantages of CL methods, including Online-LoRA, are not solely due to the use of pre-trained weights but also stem from the effectiveness of the methods themselves. UB is the upper-bound baseline trained on the i.i.d. data of the datasets. The best results are noted by **bold**.

all methods, but it is our approach that leads to superior performance.

F. Baseline Settings

In this section, we provide the experimental settings for the baseline methods used in our experiments[1].

F.1. Overview of Baselines

- **AGEM** [106]: Averaged Gradient Episodic Memory, utilizes samples in the memory buffer to constrain parameter updates.
- **ER** [107]: Experience replay, a rehearsal-based method with random sampling in memory retrieval and reservoir sampling in memory update.
- **EWC++** [105]: An online version of EWC [111], a regularization method that limits the update of parameters crucial to past tasks.
- **MIR** [102]: Maximally Interfered Retrieval, a rehearsal-based method that retrieves memory samples with loss increases given the estimated parameter update based on the current batch.
- **GDumb** [120]: Greedy Sampler and Dumb Learner, a strong baseline that greedily updates the memory buffer from the data stream with the constraint to keep a balanced class distribution.
- **PCR** [114]: Proxy-based contrastive replay, a rehearsal-based method that replaces the samples for anchor with proxies in a contrastive-based loss.
- **DER++** [104]: Dark Experience Replay++, a rehearsal-based method using knowledge distillation from past experiences.
- **LODE** [113]: Loss Decoupling, a rehearsal-based method that decouples the learning objectives of old and new tasks to minimize interference.
- **EMA** [121]: Exponential Moving Average, a model ensemble method that combines models from various training tasks.
- **L2P** [124]: Learning to Prompt, a prompt-based method that prepends learnable prompts selected from a prompt pool to the embeddings of a pre-trained transformer.
- **MVP** [119]: Mask and Visual Prompt tuning, a prompt-based method that uses instance-wise feature space masking.

F.2. Backbone

Among the baseline methods we compare, L2P [124] and MVP [119] originally reported results using a ViT-B/16 model [108] pre-trained on ImageNet21k, while the other baselines (AGEM [106], ER [107], EWC++ [105], MIR [102], GDumb [120], DER++ [104], PCR [114], LODE [113], EMA [121]) reported results using a ResNet18 [109] architecture.

To ensure a fair comparison, we standardized our experimental setup by evaluating all baselines using the same pre-trained ViT model (ViT-B/16 and ViT-S/16). For methods originally implemented with ResNet18, we reimplemented them with ViT to match the experimental conditions of L2P and MVP. As shown in Table S6, all methods perform better with the pre-trained ViT-B/16 than with ResNet18, supporting our argument that using a pre-trained ViT provides a more consistent and stronger baseline for performance comparisons.

[1]Codebases used: https://github.com/AlbinSou/online_ema.git, https://github.com/liangyanshuo/Loss-Decoupling-for-Task-Agnostic-Continual-Learning.git, https://github.com/FelixHuiweiLin/PCR.git, https://github.com/RaptorMai/online-continual-learning.git

Method	Acc. w/ ResNet18	Acc. w/ ViT-B/16	Performance Gain (%)
AGEM [106]	5.4±0.6	12.67±1.87	134.63
ER [107]	14.5±0.8	44.85±1.83	209.31
EWC++ [105]	4.8±0.2	10.61±0.74	121.04
MIR [102]	14.8±0.7	48.36±3.11	226.76
GDumb [120]	24.8±0.7	41.00±19.97	65.32
PCR [114]	21.8±0.9	48.48±0.15	122.39
DER++ [104]	15.5±1.0	36.64±6.11	136.39
LODE (DER++) [113]	37.8±1.1	44.29±1.48	17.17
EMA (DER++) [121]	23.2±1.2	42.28±4.36	82.24
EMA (RAR) [121]	35.4±1.2	47.10±0.82	33.05

Table S6. Performance comparison on CIFAR-100 between ResNet18 and pre-trained ViT-B/16 in an online class-incremental learning scenario. Acc. stands for Accuracy. All rehearsal-based methods use a buffer size of 500 for fair comparison. The results demonstrate that there is no unfair comparison in our experiments, as all methods benefit from the pre-trained ViT-B/16 model. The performance gain is computed as the percentage increase from the ResNet18 accuracy to the ViT-B/16 accuracy for each method.

F.3. Training Settings

The following settings are shared by the baseline methods (and our Online-LoRA) in the experiments:

- Buffer Size: 500. Methods using a buffer include AGEM [106], ER [107], MIR [102], GDumb [120], PCR [114], DER++ [104], LODE [113], and EMA [121].
- Optimizer: Adam.
- Batch Size: 64.

In Table S7, we summarize the hyperparameters used for all baseline methods in our experiments. To ensure a fair comparison, we adopted these hyperparameters from their original codebases. However, because the baseline methods used different backbones and batch sizes in their original experiments, we adjusted the learning rates for some baselines to standardize the comparison across all methods. For tuning the learning rates, we followed the protocol outlined in [117] and conducted a grid search on a small cross-validation set. The hyperparameter grid for the baselines is detailed in Table S8.

G. Exploration with Buffer Size

Table S9 we show more results of the impact of buffer sizes on the performance of replay-based methods (AGEM [106], ER [107], GDumb [120], MIR [102]).

As shown in Table S9, when the buffer size increases, all replay-based methods see improvements in their performance across the benchmarks. Notably, when the buffer size hits 5000 (a large capacity; 20% of the ImageNet-R training set, 12.5% of the ImageNet-S training set), the difference in performance between GDumb and other replay-based methods narrows. This suggests that the sophisticated memory retrieval strategies employed by these other methods do not significantly outperform GDumb's simple approach of training directly on the buffered data. Moreover, the performance of rehearsal-based methods drops when the buffer size shrinks. This highlights the efficiency of our Online-LoRA, which achieves high performance using just a minimal buffer size of 4.

H. Computation Analysis

In this section, we present the model parameter size, training FLOPs, and training time for our Online-LoRA and the baseline methods.

As shown in Table H, our Online-LoRA model introduces approximately 0.6M additional parameters due to the inclusion of LoRA parameters, which represents a negligible increase (0.69%) compared to the original size of the ViT-B/16 model. Notably, our memory buffer contains only 4 data samples, whereas other baselines (except EWC++) require at least 500 samples in their buffers to achieve comparable performance (see Appendix G for more details). Regarding computational consumption measured by FLOPs during training, Online-LoRA demonstrates advantages over EWC++ [105], thanks to our efficient computation of the importance weight matrix, as explained in Section 3.3 of the main paper. The extremely low FLOPs of GDumb [120] can be attributed to its design, which involves greedily updating the memory buffer without employing additional strategies. However, its training time is relatively high because retraining is triggered frequently to maintain a balanced memory buffer, which adds overhead despite the low FLOPs.

Method	CIFAR-100	ImageNet-R	ImageNet-S	CUB-200	CORe50
AGEM [106]	LR=0.0001, WD=0.0001				
ER [107]	LR=0.0001, WD=0.0001, Episode memory per batch=10				
EWC++ [105]	LR=0.0001, WD=0.0001, λ=100, α=0.9 Number of training batches after which the Fisher information will be updated: 50				
MIR [102]	LR=0.0001, WD=0.0001, Number of subsample=50				
GDumb [120]	LR=0.001, WD=0.0001, Minimal learning rate: 0.0005, Gradient clipping=10, Epochs to train for memory=30				
PCR [114]	LR=0.0001, WD=0.0001, Episode memory per batch=10, Temperature=0.09, Warmup of buffer before retrieve=4				
DER++ [104]	LR=0.0003, α=0.2, β=0.5				
LODE [113]	LR=0.0003, C=1.0, ρ=0.1				
EMA [121]	LR=0.0002, λ for warm-up: 0.9, λ=0.99				
L2P [124]	LR=0.003, Size of the prompt pool=10, Length of a single prompt=10, Number of prepended prompt=4				
MVP [119]	LR=0.005, γ=2.0, m=0.5, α=0.5				

Table S7. Hyperparameters for the baseline methods on ViT-B/16. LR: learning rate. WD: weight decay.

Method	CIFAR-100	ImageNet-R	ImageNet-S	CUB-200	CORe50
AGEM [106]	LR: [0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1] WD: [0.0001, 0.001, 0.01, 0.1]				
ER [107]	LR: [0.0001, 0.0003, 0.001, 0.003] WD: [0.0001, 0.001, 0.01, 0.1]				
EWC++ [105]	LR: [0.0001, 0.001, 0.01, 0.1] WD: [0.0001, 0.001]				
MIR [102]	LR: [0.0001, 0.001, 0.01, 0.1] WD: [0.0001, 0.001]				
GDumb [120]	LR: [0.001, 0.01, 0.1 WD: [0.0001, 0.000001]				
PCR [114]	LR: [0.0001, 0.001, 0.01, 0.1] WD: [0.0001, 0.001]				
DER++ [104]	LR: [0.0003, 0.003, 0.03]				
LODE [113]	LR: [0.0003, 0.003, 0.03]				
EMA [121]	LR: [0.0001, 0.0002, 0.0003, 0.0004, 0.0005]				

Table S8. Hyperparameter grid for the baseline methods using the ViT-B/16 backbone. LR: learning rate; WD: weight decay. Since L2P [124] and MVP [119] use the same backbone and batch size as in our experiments, their learning rates were not adjusted.

I. Task Accuracy

In this section, Figure S2 and Figure S3 show task accuracy as a function of the number of learning tasks as described in Section 4.4 in the main paper. The ViT-B/16 model is employed on the Split ImageNet-S dataset with 20 tasks. These results demonstrate that our Online-LoRA consistently outperforms the other methods in mitigating the forgetting of previously learned tasks.

Figure S2a shows that AGEM [106] begins with an ini-

Buffer size	Method	Split-ImageNet-R	Split-ImageNet-S	Core50
500	AGEM [106]	5.60±2.74	0.16±0.04	80.15±2.97
	ER [107]	40.99±3.96	30.21±0.70	85.85±1.35
	MIR [102]	41.51±2.99	30.33±3.81	74.35±4.07
	GDumb [120]	8.87±1.36	1.65±0.22	77.20±3.49
1000	AGEM [106]	7.16±1.56	0.23±0.04	78.73±3.87
	ER [107]	44.71±2.63	34.32±0.53	84.27±4.11
	MIR [102]	46.65±5.63	33.99±1.72	82.64±1.12
	GDumb [120]	19.19±1.36	2.71±0.12	78.09±3.75
5000	AGEM [106]	7.21±0.34	0.12±0.02	77.57±3.56
	ER [107]	47.23±2.71	37.65±0.23	81.32±2.19
	MIR [102]	**49.33±3.49**	35.90±2.35	81.18±3.20
	GDumb [120]	46.08±0.64	9.68±0.28	69.42±1.06
	Ours	48.18±0.63	**47.06±0.24**	**93.71±0.01**
	UB	76.78±0.44	63.82±0.02	95.60±0.01

Table S9. Results of replay-based methods with different buffer size. A_{Final} metric and ViT-B/16 model is used. Each dataset has 10 disjoint tasks. *UB* is the upper-bound baseline trained on the i.i.d. data of the datasets. The best results are noted by **bold**.

Method	#params (M)	FLOPs ($\times 10^{15}$)	Training time (s)
AGEM [106]	85.88	140.52	828.39
ER [107]	85.88	140.05	849.43
EWC++ [105]	85.88	214.36	1076.53
GDumb [120]	85.88	18.44	2078.59
MIR [102]	85.88	161.04	1069.29
Ours	86.47	151.20	864.60

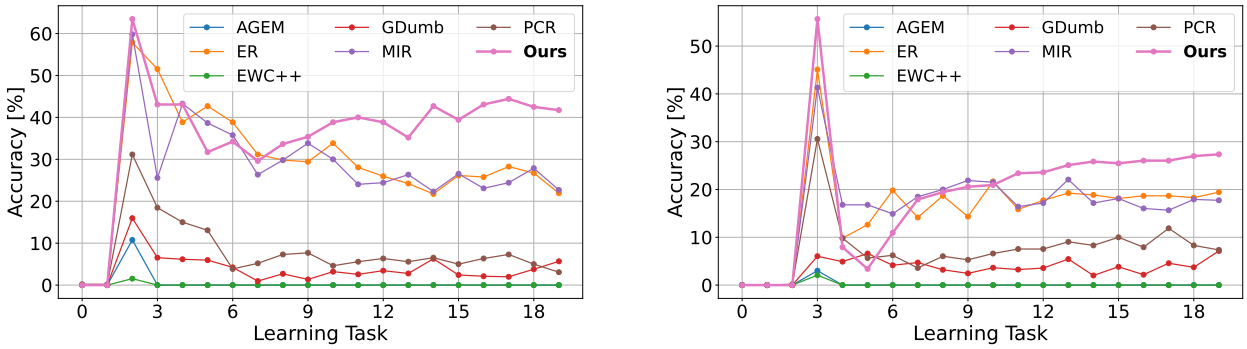
Table S10. Computational statistics for Online-LoRA and baseline methods on CIFAR-100 in the online class-incremental learning scenario using the ViT-B/16 backbone. FLOPs are measured as 'forward FLOPs per GPU' using the DeepSpeed FLOPS Profiler [118]. All experiments are conducted on a single NVIDIA A100 GPU.

tial accuracy of $\sim$10%. However, this accuracy drastically decreases for subsequent tasks, eventually dropping to zero. Given that the Split ImageNet-S dataset consists of 20 tasks with 500 classes per task, AGEM's performance is no better than that of a random model, which would have an expected accuracy of 0.2%. This dramatic decline is primarily due to the increasingly restrictive constraints placed on gradient updates as the number of tasks increases. Such constraints significantly hurt the model's ability to learn from new tasks, showing a fundamental weakness of AGEM in handling long sequences of diverse tasks. A similar issue was observed with EWC++ [105], another regularization-based approach.

In contrast, our Online-LoRA model does not encounter this problem even though an online weight regularization is used. This is because our model is continuously expanded by adding new LoRA parameters (see Section 3.2 in the main paper). This strategy allows the model to adapt to new information more flexibly, bypassing the learning limitations encountered by traditional regularization methods like AGEM and EWC++.

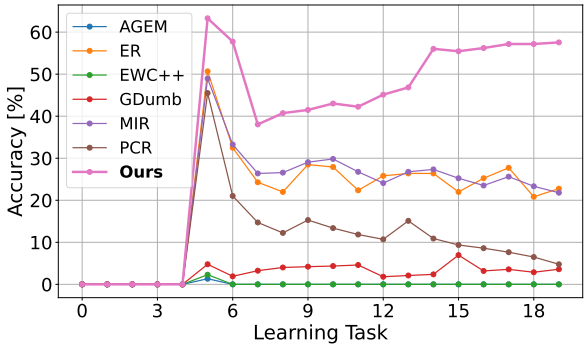
J. Code

We provide the code of our Online-LoRA as part of the supplementary materials. Our implementation of LoRA is based on the codebase of MeLo [126]. See the README file for more details.

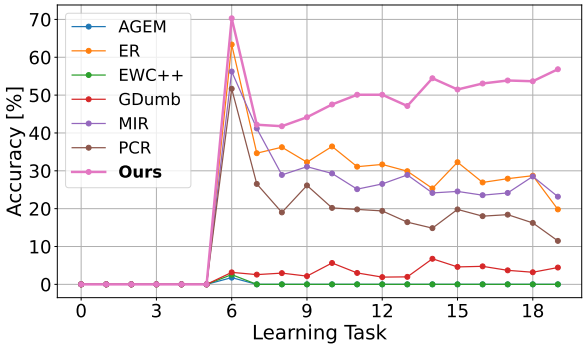


(a) Task accuracy of task #2

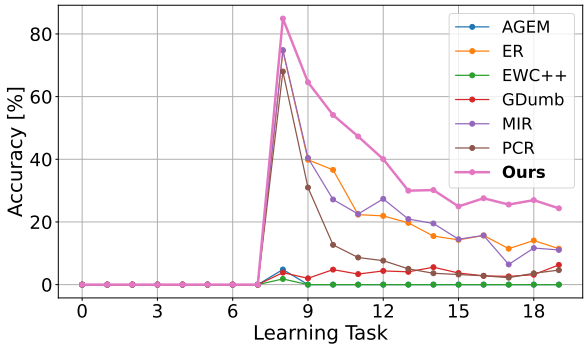
(b) Task accuracy of task #3



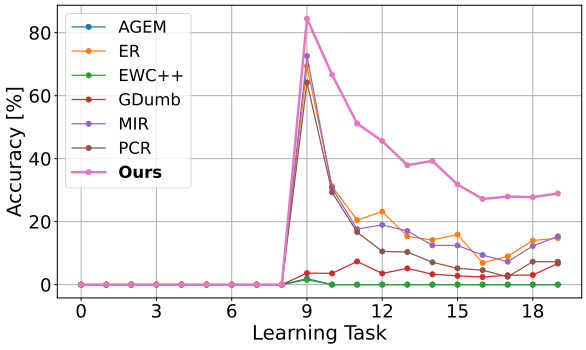
(c) Task accuracy of task #5



(d) Task accuracy of task #6

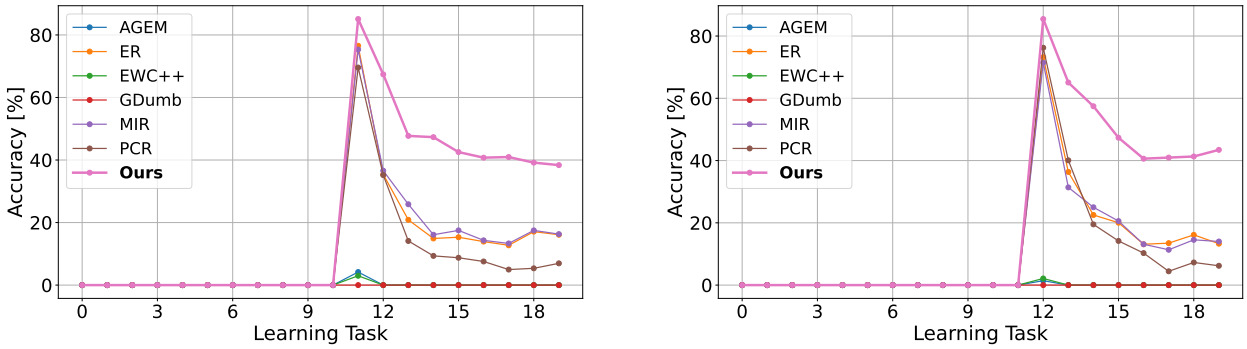


(e) Task accuracy of task #8



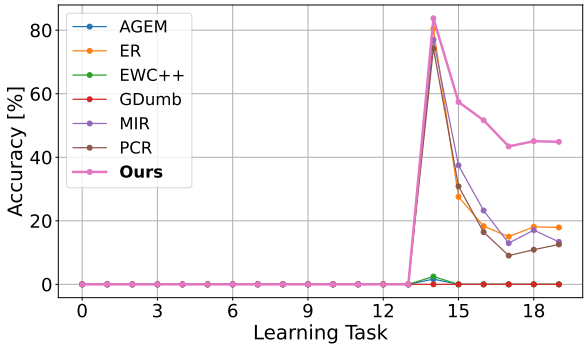
(f) Task accuracy of task #9

Figure S2. Task accuracy versus the number of learning tasks of task #2 to task #9. Our Online-LoRA consistently outperforms all the other methods in maintaining accuracy on previously learned tasks. Note that the recorded accuracy for initial tasks is zero, not due to poor model performance, but because our evaluation prioritizes mitigating forgetting in tasks the model has already encountered.

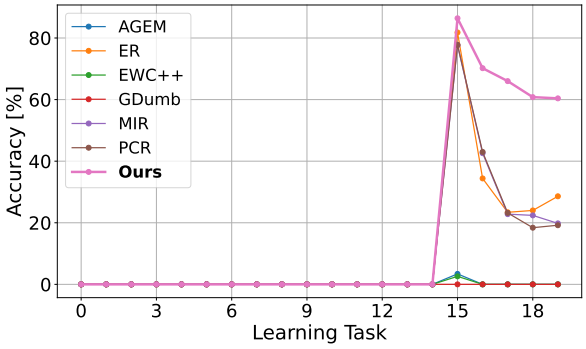


(a) Task accuracy of task #11

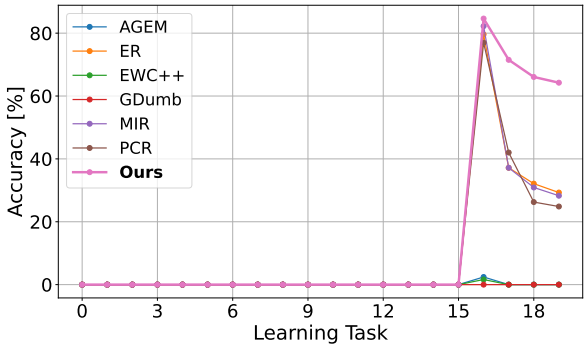
(b) Task accuracy of task #12



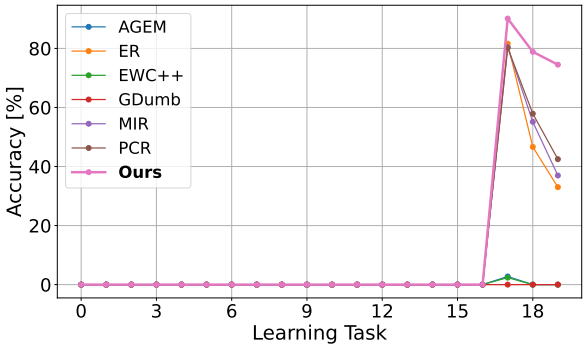
(c) Task accuracy of task #14



(d) Task accuracy of task #15



(e) Task accuracy of task #16



(f) Task accuracy of task #17

Figure S3. Task accuracy versus the number of learning tasks of task #11 to task #17. Compared to the results of task #2 to task #9 in Figure S2, our Online-LoRA has greater advantages over the other methods for these newer tasks #11 to task #17. Zero accuracy for initial tasks results from not measuring them at the time the specific task had not been learned yet.

References

[102] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019. 3, 4, 5, 6, 7

[103] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019. 1

[104] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 4, 5, 6

[105] 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)*, pages 532–547, 2018. 3, 4, 5, 6, 7

[106] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 3, 4, 5, 6, 7

[107] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019. 3, 4, 5, 6, 7

[108] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4

[109] 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*, pages 770–778, 2016. 4

[110] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 1

[111] 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. 4

[112] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 1

[113] Yan-Shuo Liang and Wu-Jun Li. Loss decoupling for task-agnostic continual learning. *Advances in Neural Information Processing Systems*, 36, 2023. 4, 5, 6

[114] Huiwei Lin, Baoquan Zhang, Shanshan Feng, Xutao Li, and Yunming Ye. Pcr: Proxy-based contrastive replay for online class-incremental continual learning. In *CVPR*, pages 24246–24255, 2023. 4, 5, 6

[115] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021. 2

[116] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pages 17–26, 2017. 1

[117] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. 1, 5

[118] Microsoft. Deepspeed: A deep learning optimization library. https://github.com/microsoft/DeepSpeed, 2024. Accessed: 2024-09-05. 7

[119] Jun-Yeong Moon, Keon-Hee Park, Jung Uk Kim, and Gyeong-Moon Park. Online class incremental learning on stochastic blurry task boundary via mask and visual prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 4, 6

[120] Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *The European Conference on Computer Vision (ECCV)*, August 2020. 3, 4, 5, 6, 7

[121] Albin Soutif-Cormerais, Antonio Carta, and Joost Van de Weijer. Improving online continual learning performance and stability with temporal ensembles. In *Conference on Lifelong Learning Agents*, pages 828–845. PMLR, 2023. 4, 5, 6

[122] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Caltech-ucsd birds 200. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1

[123] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019. 1

[124] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 4, 6

[125] Hyun woo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. *ArXiv*, abs/2110.10031, 2021. 1

[126] Yitao Zhu, Zhenrong Shen, Zihao Zhao, Sheng Wang, Xin Wang, Xiangyu Zhao, Dinggang Shen, and Qian Wang. Melo: Low-rank adaptation is better than fine-tuning for medical image diagnosis, 2023. 7