One-Shot Domain Incremental Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Domain incremental learning (DIL) has been discussed in previous studies on deep neural network models for classification. In practice, however, we may encounter a situation where we need to perform DIL under the constraint that the samples on the new domain are observed only infrequently. In this study, we consider the extreme case where we have only one sample from the new domain, which we call one-shot DIL (ODIL). In simulation experiments on ODIL, we observed that the accuracy on both the new domain and the original domain deteriorated even on applying existing DIL methods. We analyzed the reason for this problem through various investigations and discovered that the cause would be the statistics of the batch normalization layers. According to our analysis, we propose a new technique regarding these statistics and demonstrate the effectiveness of the proposed method in ODIL through experiments on open datasets.

1 Introduction

In recent years, deep learning has been widely used in image recognition, speech recognition, and natural language processing [18]. There is a need to update a trained neural network model so that the model can classify samples correctly on a new, untrained input distribution (domain) [6]. The additional training for the new domain is called domain incremental learning (DIL) [32]. In DIL, we assume that the input domains increase in each class, while the number of classes remains constant. We need to correctly classify inputs on both the new and original domains.

In practice, although previous studies on DIL assume many samples from the new domain, we often face situations in which we have few samples from the new domain.[1] Therefore, in this study, we explore one-shot DIL (ODIL), which is supposed to have only one sample from the new domain. Figure 1 shows an example of the new and original domains in ODIL, arranged using CIFAR10 [17]. In this example, no truck images are included in the original 9-class data, and such images are not used in the first training. Suppose that one truck image is given and added to the automobile class as a new domain after the first training. We assume that the model obtained by the first training misclassifies the given truck image. Therefore, we attempt to update this model with the given truck image as the second training, so that the model classifies truck images into the automobile class.

Previous studies proposed DIL methods with mechanisms to improve the accuracy on the new domain while maintaining the accuracy on the original domain. Examples of such DIL methods are Elastic Weight Consolidation (EWC) [16] and Gradient Episodic Memory (GEM) [21]. However, in our experiments simulating ODIL, we could not maintain the accuracy on the original domain even after applying EWC or GEM directly under the ODIL setting. Furthermore, though the goal of ODIL is to improve the accuracy on the new domain, it worsened in ODIL with EWC or GEM. We carefully investigated the reason for this deterioration and found that the statistics maintained in the batch normalization layers [13] are more drastically affected under the ODIL setting compared to the

[1] In Appendix B, we give an example of situations in which we have few samples from the new domain.

Submitted to the Workshop on Distribution Shifts (DistShift@NeurIPS 2023). Do not distribute.

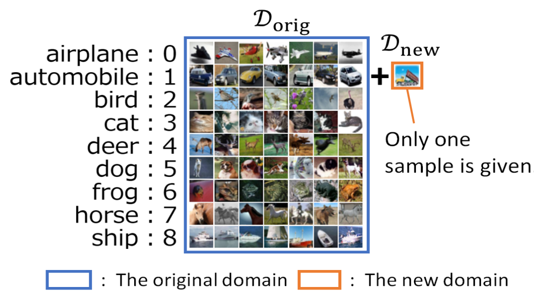


Figure 1: Example of the original domain and the new domain in one-shot domain incremental learning (ODIL) with CIFAR10 [17]. In this example, trucks are added to the "automobile" class as the new domain. However, only one sample is added.

general DIL setting. Therefore, in this paper, we propose a new technique regarding the statistics in the batch normalization layers to prevent the degradation of accuracy. We report the experimental results indicating that our technique is necessary even if we apply EWC or GEM under the ODIL setting. Note that the related work of our study is explained in Appendix A.

2 Problem Description

In this section, we define ODIL. Let $\mathcal{X}(\subseteq \mathbb{R}^d)$ be an input space. Let $\mathcal{Y} = \{1, \cdots, K\}$ be the set of classes, where $K \in \mathbb{N}$ is the number of classes. Let $\mathcal{D}_{\text{orig}} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ $(\boldsymbol{x}_n \in \mathcal{X}, y_n \in \mathcal{Y};\ n = 1, \cdots, N)$ be a labeled dataset for classification, where $\boldsymbol{x}_n \sim p_{\text{orig}}(\boldsymbol{x}|y = k)(k = 1, \cdots, K)$. In this work, $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y}$ parametrized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}(\subseteq \mathbb{R}^p)$ denotes a neural network model. We suppose that a neural network model $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}(\hat{\boldsymbol{\theta}}_{\text{orig}} \in \boldsymbol{\Theta})$ trained with $\mathcal{D}_{\text{orig}}$ is given. Under this premise, we suppose that a new labeled dataset $\mathcal{D}_{\text{new}}$ is given. We assume that $|\mathcal{D}_{\text{new}}| = 1$ and represent $\mathcal{D}_{\text{new}} = \{(\boldsymbol{x}_0, y_0)\}$ $(\boldsymbol{x}_0 \in \mathcal{X}, y_0 \in \mathcal{Y})$. The new sample $(\boldsymbol{x}_0, y_0)$ is based on the following assumptions: 1) The trained model $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ misclassifies the new sample $(\boldsymbol{x}_0, y_0)$. 2) The sample $(\boldsymbol{x}_0, y_0)$ is not included in the original dataset $\mathcal{D}_{\text{orig}}$. 3) $\boldsymbol{x}_0 \sim p_{\text{new}}(\boldsymbol{x}|y = y_0)$, where $p_{\text{orig}}(\boldsymbol{x}|y = y_0)$ and $p_{\text{new}}(\boldsymbol{x}|y = y_0)$ are not equivalent.

We define ODIL as updating the trained model $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ with $\mathcal{D}_{\text{new}}$ to obtain a new model $f_{\hat{\boldsymbol{\theta}}_{\text{new}}}(\hat{\boldsymbol{\theta}}_{\text{new}} \in \boldsymbol{\Theta})$ that can correctly classify the inputs on the new domain $p_{\text{new}}(\boldsymbol{x}|y = y_0)$. Note that the model is required to maintain the classification accuracy on the original domain $p_{\text{orig}}(\boldsymbol{x}|y = k)$ as much as possible. As in previous studies [21, 4], we allow a subset $\mathcal{M}$ of the examples in the original data $\mathcal{D}_{\text{orig}}$ to be stored. We can use the subset $\mathcal{M}$ in ODIL to prevent the deterioration of the accuracy on the original domain.

For example, in Fig. 1, the original domain contains classes other than trucks. The new domain contains only trucks. The given truck image corresponds to $\boldsymbol{x}_0$, and the label y_0 represents the automobile class. We expect that the new model $f_{\hat{\boldsymbol{\theta}}_{\text{new}}}$ will correctly classify the trucks as well as the other data. Most previous studies about DIL assumed that the domains expand in all classes. Unlike them, we assume that the domain increases in only one class because the new dataset contains only one sample.

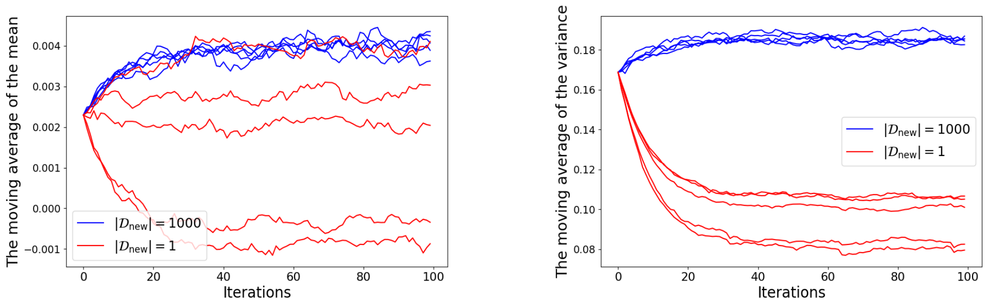
3 Method for ODIL

3.1 Problems with Batch Normalization in ODIL

EWC and GEM have mechanisms to improve accuracy on the new domain while maintaining accuracy on the original domain [16, 21]. Therefore, it is natural to expect that the same results can be achieved in ODIL as well. However, in experiments simulating ODIL, we could not maintain the accuracy on the original domain even when using EWC or GEM. Further, although ODIL was intended to improve the accuracy on the new domain, EWC and GEM actually worsened the accuracy of the new domain in ODIL, as shown in Table 1. Table 1 summarizes the test accuracy on the new

Table 1: Test accuracy on the original and new domains when $|\mathcal{D}_{\text{new}}| = 1000$ or $|\mathcal{D}_{\text{new}}| = 1$.

	$f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$	$f_{\hat{\boldsymbol{\theta}}_{\text{new}}}$					
		CE		CE+EWC		CE+GEM	
		1000	1	1000	1	1000	1
p_{new}	0.6940	0.9410	0.5765	0.9390	0.5955	0.9960	0.7315
p_{orig}	0.9550	0.9420	0.4936	0.9434	0.4647	0.9306	0.9071



(a) Transition of the moving averages of the mean. (b) Transition of the moving averages of the variance.

Figure 2: Transition of the moving averages of the statistics at the batch normalization layer closest to the input layer in ResNet18 [9]. We accumulated the moving averages of the statistics at every forward propagation and plotted their sequences. Since the input is normalized in parallel for each channel in the batch normalization layer, we computed the average for the channels.

domain $p_{\text{new}}(\boldsymbol{x}|y = y_0)$ and the original domain $p_{\text{orig}}(\boldsymbol{x}|y = k)(k = 1, \cdots, K)$ in experiments simulating ODIL as shown in Fig. 1. We tried two cases, $|\mathcal{D}_{\text{new}}| = 1$ and $|\mathcal{D}_{\text{new}}| = 1000$. In the $|\mathcal{D}_{\text{new}}| = 1$ setting, we resampled data from $\mathcal{D}_{\text{new}}$ with data augmentation [28] at each step of the optimization to increase the number of learnable samples for the new domain. The details of this data augmentation are explained in Appendix C. We used ResNet18 [9] as an image classifier. CE denotes the case where we performed optimization with simple cross-entropy loss. CE+EWC and CE+GEM are the same as CE except that EWC and GEM are applied, respectively. The setup of the experiments is explained in Appendix D. When $|\mathcal{D}_{\text{new}}| = 1$, the accuracy on both domains decreases after training with $\mathcal{D}_{\text{new}}$, in the case of CE and CE+EWC. This deterioration on both domains is a problem specific to ODIL as the deterioration on the new domain does not occur when $|\mathcal{D}_{\text{new}}| = 1000$.

We carefully analyzed the cause of this problem and discovered that the cause lies in the statistics in the batch normalization layers. Figure 2 shows the transitions of the moving averages of the mean and variance of a batch normalization layer as $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ is updated with $\mathcal{D}_{\text{new}}$. We chose an example in CE. We ran five trials with varying $\mathcal{D}_{\text{new}}$ for each of the $|\mathcal{D}_{\text{new}}| = 1000$ and $|\mathcal{D}_{\text{new}}| = 1$ settings. We plotted all results of the five trials as the five lines in the figures. In Fig. 2a, the moving average of the mean shifts in more widely different directions over the five trials in $|\mathcal{D}_{\text{new}}| = 1$ than in $|\mathcal{D}_{\text{new}}| = 1000$. When $|\mathcal{D}_{\text{new}}| = 1$, for some trials, a positive moving average is obtained, while for others, a negative moving average is obtained. This variation is caused by the fact that the mean value is calculated under a strong influence of the single input $\boldsymbol{x}_0$. The mean value heavily influenced by the single input $\boldsymbol{x}_0$ causes the trainable weights and biases in the batch normalization layers to be updated in unexpected directions during training. Furthermore, in inference, the moving average that is heavily influenced by the single input $\boldsymbol{x}_0$ degrades the generalization ability. In Fig. 2b, the moving average of the variance shifts in the opposite direction between $|\mathcal{D}_{\text{new}}| = 1000$ and $|\mathcal{D}_{\text{new}}| = 1$. The moving average of the variance increases when $|\mathcal{D}_{\text{new}}| = 1000$, which is natural from the viewpoint that the number of domains increases. However, when $|\mathcal{D}_{\text{new}}| = 1$, the moving average of the variance decreases. This discrepancy is due to a lack of diversity in the new data in ODIL. This results in a reduced variance in the mini-batches, in turn causing the trainable weights and biases in the batch normalization layers to be updated in unexpected directions during training. Furthermore, in inference, the much smaller moving average of variance causes misclassification because the outputs of the batch normalization layers become much larger than the inputs. The lower accuracies in $|\mathcal{D}_{\text{new}}| = 1$ are due to the above problems. Although we have tried EWC and GEM, existing DIL methods other than EWC and GEM are also likely to show the same trend as long as

Table 2: Test accuracy on the new and original domains in the baseline or proposed (CIFAR10)

	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{new}}}$					
		CE		CE+EWC		CE+GEM	
		baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.6940	0.5765 ±0.2297	0.9595 ±0.0430	0.5955 ±0.2614	0.9420 ±0.0586	0.7315 ±0.2287	0.9485 ±0.0417
p_{orig}	0.9550	0.4936 ±0.1847	0.9335 ±0.0440	0.4647 ±0.2202	0.9332 ±0.0192	0.9071 ±0.2512	0.9404 ±0.0250

Table 3: Test accuracy on the new and original domains in the baseline or proposed (RESISC45)

	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{new}}}$					
		CE		CE+EWC		CE+GEM	
		baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.1071	0.4250 ±0.1217	0.8250 ±0.0862	0.3893 ±0.1416	0.8214 ±0.0868	0.3643 ±0.1081	0.7786 ±0.1020
p_{orig}	0.9580	0.9482 ±0.0096	0.8912 ±0.0712	0.9502 ±0.0166	0.8879 ±0.0728	0.9478 ±0.0083	0.9294 ±0.0549

batch normalization layers are present in the model. Therefore, we propose to use fixed statistics in the batch normalization layers in ODIL, as detailed in the following section.

3.2 Modifying Batch Normalization Statistics

As explained in Section 3.1, the shifted statistics in the batch normalization layers degrade the accuracy on the new and original domains under the ODIL setting. Therefore, we modify the statistics as follows: 1) We do not update the moving averages of the statistics and we fix them to the values they had before training the new domain. 2) We use these constant moving averages for batch normalization when calculating forward and backward propagation during training. 3) We also use the constant moving averages in inference. These improvements avoid the unexpected shift of the statistics shown in Section 3.1.

4 Experiments

To investigate the effectiveness of the proposed method, we performed experiments on the image datasets CIFAR10 [17] and RESISC45 [5]. As experiments using CIFAR10, the example in Fig. 1 was adopted. In addition, as experiments using RESISC45, we adopted the example given by replacing "truck" with "airplane" and "automobile" with "airport" in Fig. 1. We used ResNet18 [9] as an image classifier. We trained the models in ODIL with the following two settings and calculated the test accuracy on both the new domain $p_{\mathrm{new}}(\boldsymbol{x}|y = y_0)$ and the original domain $p_{\mathrm{orig}}(\boldsymbol{x}|y = k)(k = 1, \cdots, K)$. **baseline:** In the batch normalization layers, the statistics are calculated from the inputs as usual. Simultaneously, the moving averages of the statistics are updated. **proposed:** As explained in Section 3.2, the batch normalization layers constantly use the moving averages obtained before training the new domain. These moving averages are not updated. Note that there is no difference between "baseline" and "proposed" other than the batch normalization layers. The setup of the experiments is explained in Appendix D. Additional experiments are shown in Appendix E.

Tables 2 and 3 list the results of CIFAR10 and RESISC45, respectively. We calculated the median and standard deviation of the accuracies over 10 trials with varying $\mathcal{D}_{\mathrm{new}}$ (median±standard deviation). In the case of CIFAR10, some settings of the baseline degraded the accuracy on both the new and original domains through ODIL. However, the proposed method avoided the deterioration of accuracy and improved the accuracy on the new domain. The proposed method improved the accuracy not only on the new domain but also on the original domain in the experiments with CIFAR10. There is no example where the proposed method is worse than the baseline in Table 2. In the case of RESISC45, the baseline improved the accuracy on the new domain through ODIL; however, compared to the baseline, the proposed method achieved higher accuracy on the new domain. Although there is a trade-off between the accuracy on the new domain and that on the original domain, the decrease in the accuracy on the original domain is much smaller than the increase in the accuracy on the new domain.

References

[1] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *Workshop on Meta-Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[2] Antreas Antoniou and Amos J Storkey. Learning to learn by self-critique. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[3] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930. Curran Associates, Inc., 2020.

[5] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[6] 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.

[7] Xiang Gao, Ripon K. Saha, Mukul R. Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1147–1158, 2020.

[8] Vibhor Gupta, Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Continual learning for multivariate time series tasks with variable input dimensions. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 161–170, 2021.

[9] 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 (CVPR)*, June 2016.

[10] Michael Hersche, Geethan Karunaratne, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. Constrained few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9057–9067, June 2022.

[11] 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 (CVPR)*, June 2019.

[12] YenChang Hsu, YenCheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *Continual Learning Workshop, 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, 2018.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, 2015.

[14] Fahdi Kanavati and Masayuki Tsuneki. Partial transfusion: on the expressive influence of trainable batch norm parameters for transfer learning. In *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*, pages 338–353, 2021.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*, 2017.

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[19] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: `http://yann.lecun.com/exdb/mnist`, Accessed on July 21, 2023*, 2, 2010.

[20] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *Workshop in International Conference on Learning Representations*, 2017.

[21] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[22] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[23] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[24] SylvestreAlvise 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 (CVPR)*, July 2017.

[25] Xuhong Ren, Bing Yu, Hua Qi, Felix Juefei-Xu, Zhuo Li, Wanli Xue, Lei Ma, and Jianjun Zhao. Few-shot guided mix for dnn repairing. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 717–721, 2020.

[26] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks*, pages 2809–2813, 2011.

[27] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4548–4557, 2018.

[28] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(60), 2019.

[29] Jeongju Sohn, Sungmin Kang, and Shin Yoo. Arachne: Search-based repair of deep neural networks. *ACM Trans. Softw. Eng. Methodol.*, 32(4), May 2023.

[30] Domen Tabernik and Danijel Skočaj. Deep learning for large-scale traffic-sign detection and recognition. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1427–1440, 2020.

[31] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[32] Gido M. van de Ven and Andreas S. Tolias. Generative replay with feedback connections as a general strategy for continual learning. arXiv preprint arXiv: 1809.10635, 2018.

[33] Moslem Yazdanpanah, Aamer Abdul Rahman, Muawiz Chaudhary, Christian Desrosiers, Mohammad Havaei, Eugene Belilovsky, and Samira Ebrahimi Kahou. Revisiting learnable affines for batch norm in few-shot transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9109–9118, June 2022.

[34] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. DeepRepair: Style-guided repairing for deep neural networks in the real-world operational environment. *IEEE Transactions on Reliability*, pages 1–16, 2021.

[35] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12455–12464, June 2021.

[36] Hao Zhang and W.K. Chan. Apricot: A weight-adaptation approach to fixing deep learning models. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 376–387, 2019.

[37] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.

[38] Stop sign. Available Online: `https://en.wikipedia.org/wiki/Stop_sign`, Accessed on January 16, 2023.

A Related Work

A.1 Continual Learning

DIL is a part of continual learning [6]. There are three types of scenarios in continual learning: DIL, class incremental learning (CIL), and task incremental learning (TIL) [32]. In DIL, several datasets with different domains are observed over time. All the datasets have the same number of classes, and a new domain is added to each class as a new dataset is observed. Both the original and new domains must be classifiable by the training model.

In CIL, each dataset that is observed over time contains an exclusive subset of the classes from a whole set of data [24, 11]. Therefore, the observation of a new dataset means the addition of new classes. The training model must classify all classes. Each time a new dataset is observed, the number of units in the output layer of neural network models is expanded for accommodating new classes. Note that this scenario assumes that different datasets have different domains.

In TIL, the output spaces of all the datasets observed over time are disjoint [27, 8]. For example, consider the case where the original dataset deals with a 10-class classification problem, while the new dataset deals with a single output regression task [12]. Neural network models often have a multi-head output layer in TIL, wherein each head is responsible for a specific task. Note that in TIL, we assume that different datasets have different domains and different output distributions.

A.2 Few-Shot Continual Learning

Recent studies on computer vision discussed few-shot CIL (FCIL) [35, 10]. FCIL is similar to CIL except that it involves a limited number of training samples. In many cases, we assume that the first training dataset has a large sample size, while the subsequent datasets have smaller sizes [31]. The new dataset has new classes that are not included in the previous datasets. A training model must discriminate between the new and original classes, even though the new classes have few samples.

Another study explored few-shot DIL [1]. The few-shot DIL called New Classes with Overwrite Settings in [1] is similar to ours. The difference lies in the number of classes in the new domain. In the earlier work [1], the number of classes in the new domain is the same as that in the original domain. In contrast, in the present work, the new domain contains only one class regardless of the number of classes in the original domain, and only one sample belongs to that one class. The meta-learning methods used in [1] assume that there are multiple classes and multiple samples in each domain. The Self-Critique and Adapt model (SCA) [2], which can achieve the best accuracy in [1], needs two datasets—a support set and a target set—for each domain. The two datasets must be different because the target set is used for validation. Hence, it is difficult to apply this method to our ODIL.

A.3 Neural Network Software Repair

In our settings, a new sample from the new domain is assumed to be misclassified before training the new domain. Therefore, we consider ODIL as the process of repairing or debugging neural network models. Some studies have discussed neural network software repair in software engineering [36, 29]. These studies aimed to reduce the number of misclassified samples by updating the weight parameters of a trained model. Note that the predictions of correctly classified samples cannot be changed when the weight parameters are updated. Some studies assumed 1000 samples as the number of samples to be repaired [34, 25]. Other studies set the number of samples to be repaired depending on the accuracy of a pre-trained model [36, 7]. Few studies discussed neural network software repair with limited training samples. In contrast, the present study considers the situation of only one additional class with only one sample. Therefore, our study is valuable not only for continual learning but also for neural network software repair.

A.4 Transfer Learning

Transfer learning [37] is similar to continual learning and neural network software repair. However, transfer learning differs from these techniques in that it allows us to forget the original data pattern. Some studies on transfer learning discussed batch normalization. Kanavati and Tsuneki [14] claimed that fine-tuning only the trainable weights and biases of the batch normalization layers yields performance similar to that achieved by fine-tuning all the weights of the neural network models. Meanwhile, Yazdanpanah et al. [33] showed that shifting and scaling normalized features by the weights and biases of the batch normalization layers is detrimental in few-shot transfer learning. They demonstrated that the removal of batch normalization weights and biases can have a positive impact on performance.

In addition, a study explored not only trainable weights and biases but also the statistics that are computed by the inputs in the batch normalization layers. Li et al. [20] achieved a deep adaptation effect to the new domain by modulating the statistics from the source domain to the target domain in the batch normalization layers. In the present work, we proposed a technique different from that used in the earlier study [20] for the statistics in the batch normalization layers. We changed the setting of statistics in the batch normalization layers depending on the number of samples on the new domain.

B Applications

Our study contributes to many cases where samples from new domains are rarely observed over time, but the model must adapt to the new domains. For example, consider the traffic sign recognition requirements of self-driving cars. For classifying traffic signs into categories such as stop, one-way, and speed limit, based on images obtained from cameras, automatic classification is made possible by using neural network models trained on a dataset containing images of traffic signs [26, 30]. However, even if we train a large image dataset of traffic signs for this classification, we may encounter a new, unprecedented type of traffic sign that will be misclassified by the trained model in the test drives. In the category of stop signs, many different types of stop signs are in use [38]. Each time we find a new type of stop sign, we must update the trained model. In particular, even if there is only one stop sign in the world that looks like the one we found, we still need to update the model so long as the self-driving cars are using it. In addition, the updated model is expected to successfully recognize the same stop sign in different situations.

C Algorithm for ODIL

Algorithm 1: Our ODIL algorithm with improved batch normalization.

Input: $\mathcal{M}$, $\mathcal{D}_{\text{new}} = \{(\boldsymbol{x}_0, y_0)\}$, $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$, $\mathcal{L}$: loss function, ν: learning rate,
  transform: the random operator of augmentation,
  $\text{judge} : \boldsymbol{\Theta} \to \{\text{True}, \text{False}\}$: the judgment for termination of iteration

Output: $f_{\hat{\boldsymbol{\theta}}_{\text{new}}}$

1 $\boldsymbol{\theta}_0 \leftarrow \hat{\boldsymbol{\theta}}_{\text{orig}}$
2 $t \leftarrow 0$
3 **while** $\text{judge}(\boldsymbol{\theta}_t) = \text{False}$ **do**
4   Sample a mini-batch $\mathcal{B}$ from $\mathcal{M}$.
5   **for** $m = 1, 2, \cdots, M$ **do**
6     $\boldsymbol{z}_m \leftarrow \text{transform}(\boldsymbol{x}_0)$
7   **end**
8   $\mathcal{C} \leftarrow \{(\boldsymbol{z}_1, y_0), (\boldsymbol{z}_2, y_0), \cdots, (\boldsymbol{z}_M, y_0)\}$
9   $\mathcal{B}' \leftarrow \mathcal{B} \cup \mathcal{C}$
10   Compute the forward and backward propagation of the loss $\mathcal{L}(\boldsymbol{\theta}_t; \mathcal{B}')$ with the proposed method in Section 3.2.
11   $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \nu \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\boldsymbol{\theta}_t; \mathcal{B}')$
12   $t \leftarrow t + 1$
13 **end**
14 $\hat{\boldsymbol{\theta}}_{\text{new}} \leftarrow \boldsymbol{\theta}_t$
15 **return** $f_{\hat{\boldsymbol{\theta}}_{\text{new}}}$

Algorithm 1 shows the details of our algorithm for ODIL. We assume that the neural network model $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ trained on the original data $\mathcal{D}_{\text{orig}}$ is given, as in Section 2. In ODIL, the new domain $p_{\text{new}}(\boldsymbol{x}|y = y_0)$ becomes the training target in addition to the original domain $p_{\text{orig}}(\boldsymbol{x}|y = k)(k = 1, \cdots, K)$. However, we have only one sample on the new domain, denoted as $(\boldsymbol{x}_0, y_0)$. This makes training on the new domain difficult. Therefore, we apply data augmentation [28] to $\boldsymbol{x}_0$, as described in Lines 5-8 in Alg. 1. We replicate the new data $\boldsymbol{x}_0$ into M samples ($M \in \mathbb{N}$) and transform them randomly and differently in each iteration.[2] The vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_M \in \mathcal{X}$ denote the new samples obtained by transforming $\boldsymbol{x}_0$. We assume that the transformed samples $\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_M$ belong to the same class y_0 as $\boldsymbol{x}_0$. However, if all the samples in the mini-batches belong to the class y_0, updating $\boldsymbol{\theta}$ with these mini-batches may result in a model that classifies all the inputs into y_0. To avoid this problem, we concatenate the mini-batch $\mathcal{B}$, which is the mini-batch of $\mathcal{M}$, with $\mathcal{C} = \{(\boldsymbol{z}_m, y_0)\}_{m=1}^{M}$ (Line 9).

Then, in forward and backward propagation (Line 10), we perform batch normalization using the technique proposed in Section 3.2. The moving averages of the statistics are not updated. Note that we did not use the proposed technique in the experiments explained in Section 3.1, and used it as "proposed" in the experiments explained in Section 4. We continue the training iteration as long as a function "judge" returns "False" and stop it when "judge" returns "True" (Line 3). We define "judge" as the function that returns "True" if the element corresponding to y_0 in the softmax output of $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$ is greater than a threshold $\delta \in [0, 1]$. "judge" returns "False" if it is less than δ.

When using EWC in general DIL, the loss for the new domain is computed only with new data [16]. In ODIL, however, we compute the loss for the new domain with $\mathcal{B}'$ in Alg. 1 to prevent the trained model from classifying all the inputs into y_0. The Fisher information needed for EWC is computed with $\mathcal{M}$. For GEM, we compute the loss for the new domain with $\mathcal{B}'$, similar to EWC. The loss for the original domain is computed with a mini-batch $\mathcal{B}$ randomly sampled from $\mathcal{M}$.

D Details of Experiments

D.1 Datasets

Although we only show the results when we used CIFAR10 and RESISC45 in the main paper, we also show the results when we used MNIST [19] in Appendix E. CIFAR10 and MNIST classify images into 10 classes, while RESISC45 classifies images into 45 classes. Each image is assigned a label representing the class to which it belongs. To extract the datasets corresponding to the subset $\mathcal{M}(\subset \mathcal{D}_{\mathrm{orig}})$ and the new data $\mathcal{D}_{\mathrm{new}} = \{(\boldsymbol{x}_0, y_0)\}$ from CIFAR10, MNIST, and RESISC45, we perform the following operations:

Step 1. Randomly split a dataset into training, validation, and test datasets, which are denoted by $\mathcal{D}^{(\mathrm{train})}$, $\mathcal{D}^{(\mathrm{val})}$, and $\mathcal{D}^{(\mathrm{test})}$, respectively.

Step 2. Exclude one common class from each of the three datasets. The excluded datasets are denoted by $\mathcal{D}_{\mathrm{new}}^{(\mathrm{train})}$, $\mathcal{D}_{\mathrm{new}}^{(\mathrm{val})}$, and $\mathcal{D}_{\mathrm{new}}^{(\mathrm{test})}$. The remaining datasets, which have 9 (or 44) classes, are denoted by $\mathcal{D}_{\mathrm{orig}}^{(\mathrm{train})}$, $\mathcal{D}_{\mathrm{orig}}^{(\mathrm{val})}$, and $\mathcal{D}_{\mathrm{orig}}^{(\mathrm{test})}$. Let y_0' be the class label of samples included in the datasets $\mathcal{D}_{\mathrm{new}}^{(\mathrm{train})}$, $\mathcal{D}_{\mathrm{new}}^{(\mathrm{val})}$, and $\mathcal{D}_{\mathrm{new}}^{(\mathrm{test})}$.

Step 3. Change y_0' to one of the remaining 9 (or 44) classes. Let y_0 be the class label after the change.

Step 4. Suppose that $\mathcal{D}_{\mathrm{orig}}^{(\mathrm{train})}$ represent the original data $\mathcal{D}_{\mathrm{orig}}$ on $p_{\mathrm{orig}}(\boldsymbol{x}|y=k)$. These data are used for training to obtain the pre-trained model $f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$. Note that $f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$ is a 9-class (or 44-class) classifier.

Step 5. Classify the samples in $\mathcal{D}_{\mathrm{new}}^{(\mathrm{train})}$ with $f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$, and extract the misclassified samples. The dataset containing the extracted samples is denoted by $\mathcal{D}'_{\mathrm{new}}$.

Step 6. Randomly select one sample from $\mathcal{D}'_{\mathrm{new}}$. We assume that the selected sample is the given new data $\mathcal{D}_{\mathrm{new}} = \{(\boldsymbol{x}_0, y_0)\}$ on $p_{\mathrm{new}}(\boldsymbol{x}|y=y_0)$.

Step 7. Randomly select 1000 samples from the dataset $\mathcal{D}_{\mathrm{orig}}^{(\mathrm{train})}$. We assume that the dataset including the selected samples is the subset $\mathcal{M}$.

Step 8. Run Alg. 1 with the given $\mathcal{M}$, $f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$, and $\mathcal{D}_{\mathrm{new}}$.

While CIFAR10 and MNIST are originally 10-class classification problems, the above process reduces the number of classes by one to a 9-class classification problem. In the case of RESISC45, the number

[2] The transformations methods are described in Appendix D

Table 4: Settings of the original and new domains in y_0.

dataset	Set 1		Set 2	
	y_0'	y_0	y_0'	y_0
CIFAR10	“truck”	“automobile"	“dog"	“cat"
MNIST	“9"	“8"	“1”	“0”
RESISC45	“airplane”	“airport”	“overpass”	“intersection”

of classes is reduced to 44. We set the new domain $p_{\text{new}}(\boldsymbol{x}|y = y_0)$ by changing y_0', which is not included in these 9 (or 44) classes, to one of these classes. Table 4 lists the settings of y_0' and y_0 in our experiments with CIFAR10, MNIST, and RESISC45. We adopted two settings per dataset, which are named Set 1 and Set 2. The results of Set 1 of CIFAR10 and RESISC45 are shown in Section 4. Other results are presented in Appendix E. In the experiments, we randomly selected 10 samples from $\mathcal{D}'_{\text{new}}$ in Step 6 and performed ODIL 10 times, assuming that each sample was the given new sample $(\boldsymbol{x}_0, y_0)$. Over the 10 trials, we calculated the median and standard deviation of the accuracy.

Besides the new data $\mathcal{D}_{\text{new}}$ and the original data $\mathcal{D}_{\text{orig}}$, the following datasets were used in the experiments.

- $\mathcal{D}^{(\text{val})}_{\text{orig}}$ is used for tuning the learning rate.
- $\mathcal{D}^{(\text{test})}_{\text{new}}$ is used to calculate the test accuracy on the new domain $p_{\text{new}}(\boldsymbol{x}|y = y_0)$.
- $\mathcal{D}^{(\text{test})}_{\text{orig}}$ is used to calculate the test accuracy on the original domain $p_{\text{orig}}(\boldsymbol{x}|y = k)$.

$\mathcal{D}^{(\text{val})}_{\text{new}}$ was not used to tune the learning rate because of the restriction that only one new data must be available for ODIL. Using the aforementioned datasets, we calculated the accuracy on the 9 (or 44) classes, excluding y_0'.

D.2 Setup for Model Training

First Training. We followed Step 1 to Step 3 in Appendix D.1 and performed the first training with only the original data $\mathcal{D}_{\text{orig}}$ in Step 4. In this training, we used SGD [3] as the optimizer. The learning rate was scheduled by CosineAnnealing [22] from 0.1 to 0.0, and there were 200 epochs. The momentum was set to 0.9 and the weight decay was set to 0.0005. We used the same setup regardless of the new data size $|\mathcal{D}_{\text{new}}|$, DIL methods (CE, CE+EWC, or CE+GEM), and batch normalization (“baseline” or “proposed”). We used the same pre-trained model $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ for all settings.

Second Training in the $|\mathcal{D}_{\text{new}}| = 1000$ setting. For the $|\mathcal{D}_{\text{new}}| = 1000$ setting, we skipped Steps 5 and 6 in Appendix D.1 and randomly selected 1000 samples from $\mathcal{D}^{(\text{train})}_{\text{new}}$. We assumed that the dataset including the selected 1000 samples is the new data $\mathcal{D}_{\text{new}}$. We extracted the subset $\mathcal{M}$ as in Step 7, and $|\mathcal{M}| = 1000$. In Step 8, we updated $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ by the Adam optimizer [15] with the fixed learning rate 10^{-5} and the number of iterations 100. The weight decay was set to 0.0001. In this setting, we did not adopt Alg. 1. We concatenated $\mathcal{D}_{\text{new}}$ and $\mathcal{M}$ prior to training iterations and performed training on the concatenated dataset using the standard approach.

Second Training in the $|\mathcal{D}_{\text{new}}| = 1$ setting. We followed Steps 5 and 6. We extracted the subset $\mathcal{M}$ as in Step 7, and $|\mathcal{M}| = 1000$. In Step 8, we updated $f_{\hat{\boldsymbol{\theta}}_{\text{orig}}}$ with the Adam optimizer. We calculated the accuracy of the validation set of the original domain for the fixed learning rate settings of $10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}$, and 10^{-1}. Then, we selected the settings that satisfied the constraint that the training iterations were terminated (“judge” returns “True”) within 100 iterations. Moreover, we adopted the learning rate with the highest accuracy among the selected settings. The weight decay was set to 0.0001. For the transformations from $\boldsymbol{x}_0$ to $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_M$ in Alg. 1, we used RandomRotation, RandomResizedCrop, RandomAffine, and RandomPerspective, which were implemented in Torchvision [23]. For “judge” in Alg. 1, we set $\delta = 0.99$. In the experiments whose results are shown in the main paper, we set $|\mathcal{B}| = |\mathcal{C}| = 32$ when running Alg. 1. In the following section, we show additional results for a different setting.

Note that we set the regularization term coefficient of EWC to 0.5 in the above two settings.

E Additional Experiments

We presented the results of Set 1 of CIFAR10 and RESISC45 in the main paper. In this section, we also show the results of Set 2 of CIFAR10 and RESISC45 and those of Sets 1 and 2 of MNIST [19]. In addition, although in the main paper, we only showed the cases in the $|\mathcal{B}| = |\mathcal{C}| = 32$ setting, in this section, we also show the cases in the $|\mathcal{B}| = 63, |\mathcal{C}| = 1$ setting.

Tables 5 and 6 show the results when we used CIFAR10. Some settings of the baseline degraded the accuracy on both the new and original domains through ODIL. However, the proposed method avoided the deterioration of accuracy and improved the accuracy on the new domain. A comparison of the proposed method with the baseline shows that the accuracy on the new domain increased by 9%-38%, and the accuracy on the original domain increased by 0%-46%. The proposed method improved the accuracy not only on the new domain but also on the original domain in the experiments with CIFAR10. There is no example where the proposed method is worse than the baseline in Table 3.

Tables 7 and 8 show the results when we used MNIST. The baseline improved the accuracy on the new domain through ODIL; however, compared to the baseline, the proposed method achieved higher accuracy on the new domain. A comparison of the proposed method with the baseline shows that the accuracy on the new domain increased by 0.8%-15% while maintaining the reduction in accuracy on the original domain below 0.6%. Although there is a trade-off between the accuracy on the new domain and that on the original domain, the decrease in the accuracy on the original domain is much smaller than the increase in the accuracy on the new domain.

RESISC45 results show similar trends to MNIST. From Tables 9 and 10, a comparison of the proposed method with the baseline shows that the accuracy on the new domain increased by 37%-57% while maintaining the reduction in accuracy on the original domain below 7%. Similar to MNIST, the decrease in the accuracy on the original domain is much smaller than the increase in the accuracy on the new domain.

Note that the difference between the baseline and the proposed method lies only in the statistics of the batch normalization layers. These results show that good accuracy can be achieved just by modifying the batch normalization statistics. As explained in Section 3.1, the statistics shift unexpectedly in the baseline, and avoiding this unexpected shift has a strong effect on the accuracy in ODIL. Moreover, from the results of CE, we can predict that other DIL methods with cross-entropy loss show the same trend.

Table 5: Test accuracy on the new and original domains in the baseline or proposed method (Set 1 of CIFAR10)

domain	$f_{\hat{\theta}_{\text{orig}}}$	batch size	$f_{\hat{\theta}_{\text{new}}}$					
			CE		CE+EWC		CE+GEM	
			baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.6940	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.5765 ±0.2297	0.9595 ±0.0430	0.5955 ±0.2614	0.9420 ±0.0586	0.7315 ±0.2287	0.9485 ±0.0417
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.6895 ±0.1466	0.9260 ±0.0425	0.7815 ±0.1008	0.9375 ±0.0456	0.8015 ±0.0901	0.9220 ±0.0452
p_{orig}	0.9550	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.4936 ±0.1847	0.9335 ±0.0440	0.4647 ±0.2202	0.9332 ±0.0192	0.9071 ±0.2512	0.9404 ±0.0250
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9413 ±0.0054	0.9424 ±0.0091	0.9394 ±0.0070	0.9414 ±0.0164	0.9399 ±0.0062	0.9431 ±0.0137

Table 6: Test accuracy on the new and original domains in the baseline or proposed method (Set 2 of CIFAR10)

domain	$f_{\hat{\theta}_{\text{orig}}}$	batch size	$f_{\hat{\theta}_{\text{new}}}$					
			CE		CE+EWC		CE+GEM	
			baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.7800	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.7595 ±0.3077	0.9335 ±0.0257	0.7665 ±0.2721	0.9330 ±0.0241	0.8081 ±0.1010	0.9240 ±0.0255
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.7740 ±0.0771	0.9050 ±0.0110	0.7305 ±0.0526	0.9035 ±0.0260	0.8075 ±0.0663	0.8985 ±0.0223
p_{orig}	0.9672	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.9363 ±0.2308	0.9528 ±0.0267	0.9360 ±0.2210	0.9522 ±0.0259	0.9556 ±0.0313	0.9556 ±0.0201
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9544 ±0.0051	0.9580 ±0.0023	0.9580 ±0.0057	0.9587 ±0.0204	0.9532 ±0.0051	0.9584 ±0.0041

Table 7: Test accuracy on the new and original domains in the baseline or proposed method (Set 1 of MNIST)

domain	$f_{\hat{\theta}_{\text{orig}}}$	batch size	$f_{\hat{\theta}_{\text{new}}}$					
			CE		CE+EWC		CE+GEM	
			baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.1021	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.7671 ±0.1880	0.8826 ±0.0715	0.8414 ±0.1454	0.8930 ±0.0611	0.7800 ±0.1108	0.8677 ±0.0688
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.7522 ±0.2107	0.8251 ±0.0832	0.7190 ± 0.1416	0.8746 ±0.0778	0.7507 ±0.1017	0.8593 ±0.0863
p_{orig}	0.9962	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.9940 ±0.0203	0.9904 ±0.0046	0.9953 ±0.0068	0.9903 ±0.0050	0.9958 ±0.0043	0.9909 ±0.0035
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9959 ±0.2806	0.9921 ± 0.0026	0.9959 ±0.0013	0.9917 ±0.0022	0.9957 ±0.0019	0.9928 ±0.0047

Table 8: Test accuracy on the new and original domains in the baseline or proposed method (Set 2 of MNIST)

domain	$f_{\hat{\theta}_{\text{orig}}}$	batch size	$f_{\hat{\theta}_{\text{new}}}$					
			CE		CE+EWC		CE+GEM	
			baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.0555	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.8639 ±0.0763	0.9313 ±0.1016	0.8656 ± 0.1085	0.9256 ±0.0972	0.8819 ±0.1008	0.9198 ±0.0968
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9110 ±0.0692	0.9194 ±0.1089	0.8863 ±0.0847	0.9167 ±0.0783	0.8903 ±0.0751	0.9022 ± 0.0676
p_{orig}	0.9951	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.9933 ±0.0049	0.9931 ±0.0056	0.9931 ±0.0377	0.9932 ±0.0055	0.9936 ±0.0015	0.9937 ±0.0026
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9946 ±0.0005	0.9936 ±0.0010	0.9941 ±0.0004	0.9937 ±0.0026	0.9941 ±0.0011	0.9941 ±0.0008

Table 9: Test accuracy on the new and original domains in the baseline or proposed method (Set 1 of RESISC45)

domain	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$	batch size	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{new}}}$					
			CE		CE+EWC		CE+GEM	
			baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.1071	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.4250 $\pm$0.1217	0.8250 $\pm$0.0862	0.3893 $\pm$0.1416	0.8214 $\pm$0.0868	0.3643 $\pm$0.1081	0.7786 $\pm$0.1020
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.2750 $\pm$0.1067	0.7179 $\pm$0.1433	0.2750 $\pm$0.0637	0.7179 $\pm$0.1433	0.3321 $\pm$0.1187	0.7393 $\pm$0.1559
p_{orig}	0.9580	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.9482 $\pm$0.0096	0.8912 $\pm$0.0712	0.9502 $\pm$0.0166	0.8879 $\pm$0.0728	0.9478 $\pm$0.0083	0.9294 $\pm$0.0549
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9515 $\pm$0.0117	0.9403 $\pm$0.0117	0.9539 $\pm$0.0023	0.9402 $\pm$0.0117	0.9546 $\pm$0.0231	0.9410 $\pm$0.0124

Table 10: Test accuracy on the new and original domains in the baseline or proposed method (Set 2 of RESISC45)

domain	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{orig}}}$	batch size	$f_{\hat{\boldsymbol{\theta}}_{\mathrm{new}}}$					
			CE		CE+EWC		CE+GEM	
			baseline	proposed	baseline	proposed	baseline	proposed
p_{new}	0.0429	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.1500 $\pm$0.0584	0.7179 $\pm$0.1591	0.1393 $\pm$0.0776	0.7179 $\pm$0.1591	0.1429 $\pm$0.0616	0.5786 $\pm$0.1512
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.0429 $\pm$0.0450	0.5250 $\pm$0.1475	0.0500 $\pm$0.0421	0.5250 $\pm$0.1475	0.0929 $\pm$0.0463	0.4714 $\pm$0.1187
p_{orig}	0.9584	$\lvert\mathcal{B}\rvert = \lvert\mathcal{C}\rvert = 32$	0.9505 $\pm$0.0105	0.9265 $\pm$0.0455	0.9481 $\pm$0.0118	0.9265 $\pm$0.0456	0.9499 $\pm$0.0108	0.9356 0.0217
		$\lvert\mathcal{B}\rvert = 63, \lvert\mathcal{C}\rvert = 1$	0.9535 $\pm$0.0025	0.9388 $\pm$0.0172	0.9543 $\pm$0.0022	0.9389 $\pm$0.0171	0.9563 $\pm$0.0006	0.9425 $\pm$0.0113