# CLUE: Consolidating Learned and Undergoing Experience in Domain-Incremental Classification

Chengyi Cai[1], Jiaxin Liu[1], Wendi Yu[1], and Yuchen Guo[2⋆]

[1] Tsinghua-Berkeley Shenzhen Institute,
[2] Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing 100084, China
{ccy20, liujiaxi20, ywd20}@mails.tsinghua.edu.cn
yuchen.w.guo@gmail.com

**Abstract.** Deep neural networks tend to be vulnerable to catastrophic forgetting when learning new tasks. To address it, continual learning has become a promising and popular research field in recent years. It is noticed that plentiful research predominantly focuses on class-incremental (CI) settings. However, another practical setting, domain-incremental (DI) learning, where the domain distribution shifts in new tasks, also suffers from deteriorating rigidity and should be emphasized. Concentrating on the DI setting, in which the learned model is overwritten by new domains and is no longer valid for former tasks, a novel method named Consolidating Learned and Undergoing Experience (CLUE) is proposed in this paper. In particular, CLUE consolidates former and current experiences by setting penalties on feature extractor distortion and sample outputs alteration. CLUE is highly applicable to classification models as neither extra parameters nor processing steps are introduced. It is observed through extensive experiments that CLUE achieves significant performance improvement compared with other baselines in the three benchmarks. In addition, CLUE is robust even with fewer replay samples. Moreover, its feasibility is supported by both theoretical derivation and model interpretability visualization.[3]

## 1 Introduction

Humans are born with the ability to learn continuously, adapting to new scenarios without forgetting previous knowledge when facing the ever-changing world. However, when deep neural networks are applied in learning new tasks, sharp declines will be observed in the performance of previous ones, which is called catastrophic forgetting[13]. Since multiple new training samples, unpredictable scenario changes, and novel requirements in new tasks keep emerging as time passes by, making it impractical to seek a once-for-all training scheme[37, 25, 1], alleviating forgetting in continual learning[37, 25] is of great significance.

⋆ Corresponding author
[3] The code is available at: https://github.com/Multiplied-by-1/CLUE

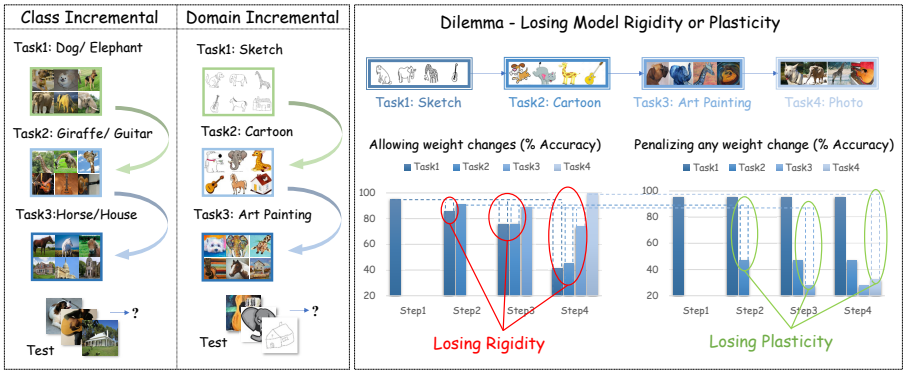


**Fig. 1.** (a) Difference between CI and DI learning. CI learning shares all the domains and divides the classes into tasks, while DI learning shares all the classes and possess different domains in each task. (b) DI learning also faces a tradeoff between rigidity and plasticity - allowing weights to be accessible to all changes leads to forgetting while penalizing and forbidding any weight change results in failure of learning new tasks.

Though abundant works have focused on continual learning[37, 25, 23, 20, 6], the majority only consider class-incremental (CI) scenario[23, 20], where different classes are distributed into sequentially-appearing tasks. Nevertheless, in real cases, sometimes it is the domain distribution that changes in the incoming tasks instead of the classes. For example, regarding autonomous driving, though the target objects to classify, such as cars, cyclists, and pedestrians, are always the same, the domain may shift due to different weather, location, and time. Thus, though being ignored currently, domain-incremental (DI)[37] learning is also an important and urgent research topic. The difference between CI continual learning and DI continual learning is depicted in Figure 1(a).

Besides, the same as its CI counterpart, DI classification also faces the rigidity-and-plasticity delimma[7]. Namely, for continual learning methods, allowing enough plasticity might result in catastrophic forgetting, while allowing no degradation in previous tasks may lead to the incapability of transferring to new tasks, which is illustrated in Figure 1(b).

Since the up-to-date research in domain-incremental continual learning[35, 41, 38, 34] does not have a unified problem setting, we first clarify the problem setting following the principles in [37]. Then, to find the main culprit of catastrophic forgetting in DI settings, we conduct preliminary experiments to observe the performance of deep learning models facing sequential tasks. As is shown in Figure 2, when learning novel tasks, the changes in data distribution lead to an alteration of the original feature extractor, which further creates a distorted attention map and invalid logits (output before Softmax) of previous data, contributing to the misclassification.

In order to alleviate forgetting, the feature extractor updated with data in novel distribution should remain valid for previous data. Therefore, a Consolidat-

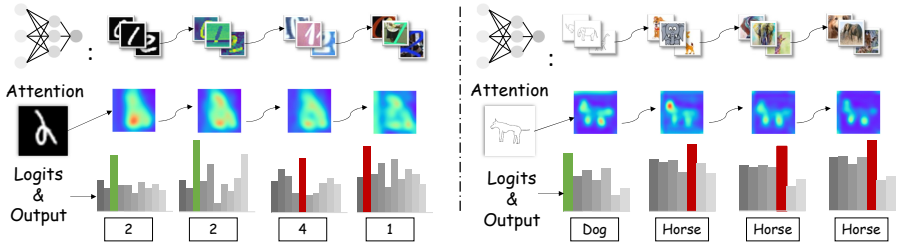


**Fig. 2.** Results of preliminary experiments. Facing novel tasks, the changes in domain distribution lead to distorted attention maps and invalid logits outputs, contributing to the misclassification. *Left* shows an example in the Digits benchmark while *right* shows an example in the Pictures benchmark.

ing Learned and Undergoing Experience (CLUE) method is proposed to maintain the performance of old tasks when training on novel ones. CLUE can be applied to all deep classification models and ensure an unchanged network architecture without additional calculation overhead. In CLUE, a network knowledge distillation loss and a stored data regularization loss are respectively utilized to increase the rigidity of the feature extractor and maintain the logits output distribution. CLUE greatly reduces forgetting and achieves good performance compared to baselines on three domain-incremental benchmarks - Digits, Pictures, Processing - we arrange in this paper. Besides, it shows remarkable robustness towards changes in the buffer size (numbers of stored replay samples), which greatly lessen storage overhead. Moreover, the feasibility of CLUE is supported by mathematical derivation and neural network interpretability visualization. Our main contributions are as follows:

- We clarify the problem setting of domain-incremental (DI) classification, define the metrics, and find the primary cause of forgetting in DI settings.
- For DI settings, we propose a Consolidating Learned and Undergoing Experience (CLUE) method, which consolidates knowledge by two additional losses from network and data aspects without introducing extra parameters or processing steps, and can be applied to mainstream deep classifiers.
- CLUE achieves remarkable performance compared with other baselines on three benchmarks - Digits, Pictures, and Processing - from different domain-incremental aspects, which are arranged using existing open-sourced datasets.
- Further experiments show that CLUE also possesses strong robustness when buffer size changes. Besides, the feasibility of CLUE is also supported by theoretical derivation and model interpretability visualization.

## 2 Related Work

### Continual Learning

An increasing amount of research[23, 20] tries to alleviate forgetting in continual learning settings. Regarding image classification, the most common mainstream method is the replay-based method [23, 20, 6]. The simple-yet-efficient method store samples from previous tasks and merge them with data of novel tasks[7]. Recent years has witnessed a great evolution of replay-based methods, from storing raw samples[3, 19, 27, 4] to synthetic image generators[33], intermediate features[40] or hidden representations[36], compressed embeddings[10] and prototypes[45]. Other non-replay methods may set some regularization. By attaching a term to the loss function to prevent weights from unwanted changes[14, 44], orthogonally modifying weights[43] or applying knowledge distillation[17] to maintain the performance, those methods help to avoid forgetting. Another typical method is to redesign the architecture into a growable network[31, 42] or apply a masked network[22, 8, 21, 32] to allow training of novel classes. However, those architecture-based methods tend to be efficient only when the task label is available in reference[37, 20, 6].

Nevertheless, the above research mainly focuses on class-incremental classification tasks, where novel classes following the same domain distribution will appear in new tasks. In this paper, we concentrate on the domain-incremental setting and propose our method to avoid catastrophic forgetting depending on why performance drops facing newly appeared alien domains.

**Continual Domain Adaptation**

Domain adaptation[26, 39], where training data and target test data are from different domains, has been another popular research topic. Nevertheless, it only prioritizes performance in the target domain without caring about catastrophic forgetting, which is partially different from continual learning settings. Some recent work begins to handle forgetting in continual domain adaptation. However, the up-to-date research[35, 41, 38, 34] does not have a unified problem setting. [34] focuses on class-incremental learning when cross-domain training sets are available. In [41], both domains and classes will increment in future tasks. Instead of sequentially appearing tasks, the source and target domains' performance is focused in [35]. Besides, most of them are based on heavy image preprocessing[38] or expandable network[41, 34] structure, which introduces additional computational overhead and may be invalid for some specific deep learning classifiers and practical application.

Therefore, this paper simplifies and clarifies the definition of DI image classification as stated in [37]. Besides, we propose a continual learning method without additional training overhead, such as image pre-processing or meta-learning calculations, and ensure that the network architecture remains unchanged facing novel tasks, which is applicable for all deep learning classifiers.

# 3 Method

## 3.1 Problem Statement

Assuming $\mathbb{D}_{train} = \{\mathbf{D}_t\}_{t=1}^{T}$ is the domain-incremental classfication training dataset with $T$ tasks, each task satisfies $\mathbf{D}_t = \{(I_{t,i}, c_{t,i})\}_{i=1}^{|\mathbf{D}_t|}$ with $|\mathbf{D}_t|$ samples. In a training image sample of task $t$, $(I_{t,i}, c_{t,i})$, $I_{t,i}$ represents for the $i^{th}$ image, and $c_{t,i}$ is its class label. Categories are shared in all tasks. Namely, $c_{t,i} \in \mathbb{C}, t = 1, 2, ..., T$, where $\mathbb{C}$ is the set of all classes. Besides, each task $t$ belongs to one specific domain $d$ without overlapping with others.

While training on task $t$, only current data $\mathbf{D}_t$ is available. Besides, a small buffer $\mathbf{B} = \{(I_b, info)\}_{b=1}^{|\mathbf{B}|}$ is also allowed to store former samples and corresponding information, where the buffer size $|\mathbf{B}|$ should be far less than $|\mathbf{D}_t|, t = 1, 2, ..., T$.

**The Goal of DI Classification**

The optimizing goal for learning task $t$ is to minimize the loss of classifying all classes in learned and undergoing tasks when only $\mathbf{D_t} \cap \mathbf{B}$ is available, which can be presented as:

$$\min[\sum_{j=1}^{|\mathbf{D}_i|} L_{CE}(F(\mathbf{\Theta}_t; I_{i,j}), c_{i,j})]_{i=1}^{t} \tag{1}$$

Where $L_{CE}$ represents the cross-entropy loss function, $F$ represents forwarding propagation calculation, and $\mathbf{\Theta}_t$ is the current model.

During reference, $\mathbb{D}_{test}$ is used to evaluate the performance, where all the domains $\{d_1, d_2, ..., d_t\}$ are included.

**Metrics of DI classification**

In the DI continual learning setting, task-average accuracy and Forgetting are the evaluation metrics.

In realistic situations, sample numbers might vary between domains because data in a particular domain (e.g., hand-written digits) might be easier to collect. To avoid overlooking the performance in some scenes with fewer test samples, we calculate the mean classification accuracy of every task, which can be formulated as:

$$\mathbf{ACC}_{avg} = \frac{1}{T} \sum_{t=1}^{T} ACC(t, T) \tag{2}$$

Where $ACC(t, T)$ means the classification accuracy of task $t$ after learning the $T^{th}$ task, the end of continual learning.

Forgetting is another commonly used continual learning metric reflecting the severity of forgetting[4]. However, only measuring the absolute accuracy drops in DI settings might be unreasonable since learning difficulty varies between domains. For example, training hand-written digits might achieve a higher accuracy

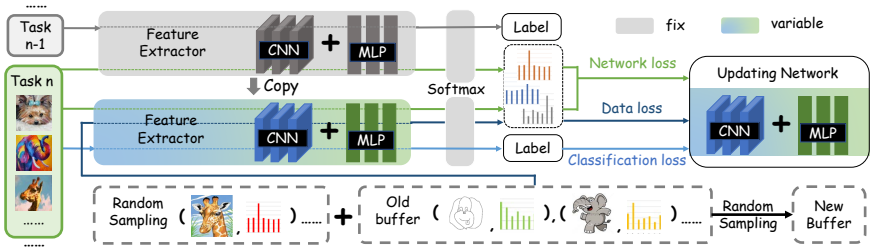


**Fig. 3.** The framework of CLUE. Replay images and corresponding logits are stored in the fixed memory. Network loss and Data loss are utilized to maintain the rigidity of feature extractor to consolidate the learned and undergoing experiences.

than digits in street scenes. Thus, the absolute accuracy decay when catastrophic forgetting happens might be more significant in hand-written digits. Therefore, ensuring the balance between domains, we also calculate task-average forgetting:

$$\mathbf{F}_{avg} = \frac{1}{T}\sum_{t=1}^{T}\frac{\max(ACC(t,t) - ACC(t,T), 0)}{ACC(t,t)} \tag{3}$$

### 3.2 Consolidating Learned and Undergoing Experience (CLUE)

**Motivation and Framework**

As shown in Figure 2, when a deep neural network faces sequential tasks from alien domain distributions, the weights in the original model adapt themselves to classify samples following the new domain distribution. The overwritten weights then lead to distorted attention maps of former task samples, which results in the invalid logits output and incorrect classification. Figure 2 shows how the handwritten digit '2' is misclassified into '4' and '1' after learning street scene and synthetic numbers because of no-more appropriate feature extractor and how the sketch image of 'Dog' is classified into 'Horse' after training on comic, painting and photos.

Thus, the main reason for forgetting tends to differ from that of the CI setting, where unbalanced, fully connected layers contribute most to forgetting. In this paper, focusing on DI problems, we propose CLUE that maintains the feature extractor's performance on previous tasks.

Figure 3 shows the overall framework of CLUE. Two loss functions concerning the original network and data are used to consolidate the learned and undergoing tasks, apart from the common cross-entropy classification loss. The training for each task requires two steps - updating the model and refreshing the buffer. In the first step, a network loss is calculated as the knowledge distillation between former and updated networks using current data. On the other hand, a data loss controls the logits alteration of stored images in the buffer. With both of the loss functions, the model is updated. In the second step, the buffer is renewed with random sampling, where the allowed account of stored images is equally

arranged for each task. Old samples are randomly dropped from the buffer if it is full, while new ones are selected from the current task to fill the vacancy. Both the image and its logits are stored.

**Loss of Learned Network**

Since $(I_{i,j}, c_{i,j}) \sim \mathbf{D}_i, i = 1, 2, ..., t-1$ in equation (1) is not available when learning task $t$, we need to estimate the optimization goal in (1) using available data. Knowledge distillation loss[12] can be used to measure the changes when training the model. Thus, the feature extractor can be prevented from becoming invalid in former tasks. Utilizing data in current tasks, we can calculate the model's output before updating and compare it with the output after training the current task. The final network loss applying knowledge distillation can be written as:

$$L_{net} = -\sum_{j=1}^{|\mathbf{D}_t|}(F(\mathbf{\Theta}_{t-1}; I_{t,j})/Tem)\log(F(\mathbf{\Theta}_t; I_{t,j})/Tem) \tag{4}$$

Where $|\mathbf{D}_t|$ represents the number of samples in current task $t$, $\mathbf{\Theta}_{t-1}$ and $\mathbf{\Theta}_t$ respectively represents the model before and after training task $t$, and $Tem \geq 1$ is the tempurate that rescale the output and is set to be 2 following [17] in this paper.

The supplement will deliver a detailed formula derivation about why $L_{net}$ estimate equation (1) and can be used in optimization.

**Loss of Learned Samples**

Another method to estimate equation (1) is to take advantage of the stored samples in the replay buffer. The replay samples directly guarantee the model performance on old tasks. To keep the output logits the same as when the replay sample is trained, we minimize the distance between logits of the current model and stored logits. The data loss $L_{data}$ can be represented as:

$$L_{data} = \sum_{b=1}^{|\mathbf{B}|} ||F(\mathbf{\Theta}_t; I_b) - z_b||_2^2 \tag{5}$$

Where $|\mathbf{B}|$ is the buffer size, $I_b$ represents the stored image, and $z_b$ represents its corresponding stored logits.

$L2$ norm is used instead of the cross-entropy loss, whose effectiveness is further discussed in the ablation study. Besides, please refer to the supplement for a detailed theoretical derivation of how $L_{data}$ relates to equation (1).

**Overall Loss Function**

Attaching $L_{net}$ and $L_{data}$ to the training loss of current classification task $L_{CE}$, we will get the overall loss function to update training models under the CLUE framework:

$$L_{all} = L_{CE} + \lambda_1 L_{net} + \lambda_2 L_{data} \tag{6}$$

Where $\lambda_1$ and $\lambda_2$ are both hyper-parameters balancing different losses.

## 4 Results

**Benchmarks**

The following three benchmarks are used in this paper:

**Digits.** We organize commonly-used digit datasets as the Digits benchmark, sorting them from simple to complex. The four tasks appear sequentially in the order of MNIST[15], MNIST-M[9], SVHN[24], Synthesis[30] to test the continual learning ability of methods.

**Pictures.** PACS dataset[16] contains object images of various types, from line sketches to real pictures, where the object categories are shared between domains. We also arrange four tasks orderly from simple to complex. Sketch, cartoons, art paintings, and authentic photo images appear sequentially.

**Processing.** STL10[5] contains ten categories of animal images. Here, we perform image processing on STL10 to divide four different types of domains. We use PyTorch's official transform tool[28] for image processing and set four sequential tasks for brightness, grayscale, sharpness, and contrast changes.

Through benchmarks from the three different aspects, we investigate the performance of methods for continual learning in the DI settings. Please refer to the supplement for more details concerning benchmarks.

**Compared Baselines**

We first implement two basic methods as upper and lower bounds for performance comparison in continual learning. *Naive*, which means that all weights in the original classifier are variable for finetuning facing novel tasks, without adding any continual learning approach, is considered the theoretical lower bound. *Joint training*, where it is assumed that data of all tasks are available simultaneously to train the model together, is considered the theoretical upper bound. It is worth noting that they are not actual experimental upper and lower bounds. In some cases, continual learning methods may exceed the range. For other comparison methods, we implement *LwF*[17], *EWC*[14], and *SI*[44] three regularization-based method, where knowledge distillation or weight update methods are utilized to control network changes. With regard to replay-based methods, *Replay*[29], *DER*[2], *DER++*[2] are implemented where replaying raw samples and feature vectors are both included.

**Implement Details**

All experiments are conducted on RTX 3090 GPUs. We use the same classifier backbone to ensure fair comparisons between methods and implement all methods in the Avalanche[18] continual learning framework with PyTorch[28].

ResNet-18[11] is used for all tasks. Besides, we use an SGD optimizer with a learning rate of 0.01 and a momentum of 0.9 for all experiments. The batch size for the Digits benchmark is 128, while the Pictures and Processing benchmark

**Table 1.** Performance Comparision on Different Benchmarks (%)

| | Benchmark 1: Digits | Benchmark 2: Pictures | Benchmark 3: Processing |
|---|---|---|---|
| Matrics | $\mathbf{ACC_{avg}} \uparrow$ | $\mathbf{ACC_{avg}} \uparrow$ | $\mathbf{ACC_{avg}} \uparrow$ |
| Naïve | 64.52 ± 0.25 | 66.91 ± 1.82 | 68.88 ± 0.54 |
| LwF | 71.89 ± 1.26 | 84.03 ± 0.75 | 73.98 ± 1.35 |
| EWC | 62.28 ± 3.06 | 69.80 ± 2.78 | 69.98 ± 1.20 |
| SI | 64.60 ± 2.22 | 69.65 ± 2.96 | 66.47 ± 1.82 |
| Replay | 78.73 ± 0.54 | 87.32 ± 0.64 | 77.52 ± 0.73 |
| DER | 79.37 ± 0.53 | 87.84 ± 0.99 | 75.58 ± 1.02 |
| DER++ | 80.27 ± 0.93 | 88.68 ± 0.28 | 75.28 ± 1.01 |
| **Ours** | **84.81 ± 0.31** | **90.16 ± 0.82** | **79.13 ± 0.98** |
| Joint | 90.67 ± 0.14 | 88.30 ± 1.70 | 88.12 ± 0.41 |
| Matrics | $\mathbf{F_{avg}} \downarrow$ | $\mathbf{F_{avg}} \downarrow$ | $\mathbf{F_{avg}} \downarrow$ |
| Naïve | 42.24 ± 0.24 | 38.24 ± 2.45 | 23.29 ± 0.57 |
| LwF | 31.86 ± 1.88 | 13.56 ± 1.58 | 18.18 ± 1.49 |
| EWC | 44.97 ± 4.56 | 34.61 ± 4.15 | 21.85 ± 1.83 |
| SI | 42.32 ± 3.45 | 34.21 ± 4.33 | 27.38 ± 2.85 |
| Replay | 22.91 ± 0.72 | 9.36 ± 0.70 | 6.80 ± 2.04 |
| DER | 20.52 ± 0.71 | 3.77 ± 0.55 | 6.61 ± 2.22 |
| DER++ | 19.23 ± 1.57 | 3.95 ± 1.05 | 8.28 ± 1.93 |
| **Ours** | **7.23 ± 0.44** | **1.34 ± 0.18** | **5.03 ± 1.21** |
| Joint | - | - | - |

have a batch size of 64. Training epochs are set to be 20 for Digits and Pictures and 40 for Processing.

To determine the hyperparameters $\lambda_1$ and $\lambda_2$ in our method, we use the grid parameter tuning method. $\lambda_1$ is determined to be 0.8, 0.2, and 0.5 for the Digits, Pictures, Processing benchmark. $\lambda_2$ is decided to be 0.1 for all benchmarks.

### 4.1 Performance Comparison

Results are indicated in Table 1 and Figure 4. Table 1 shows the metric results of different methods on the three benchmarks. Besides, the dynamic change plot of overall accuracy, which is different from the task-average accuracy $ACC_{avg}$, is depicted in the left part of Figure 4. The eventual accuracy of all tasks after learning the last one is plotted in the right part of Figure 4.

Table 1 shows that CLUE exceeds other methods by a large margin, with 84.81%, 90.16%, and 79.13% of $ACC_{avg}$ in each benchmark. It is remarkable that in the Picture benchmark, CLUE even surpasses the upper bound Joint. It is probably because the obvious difference between different domains confuses the neural network when training them jointly. However, the neural network can be trained on single novel tasks without confusion carrying former knowledge by applying CLUE. CLUE is also superb in solving catastrophic forgetting, with only 7.23% forgetting compared with 19.23% of the second-best method in the Digits

benchmark. In the Picture and Processing benchmarks, CLUE is also excellent in alleviating forgetting, only having 1.34% and 5.03% $F_{avg}$ respectively.

Besides, CLUE also maintains a high overall classification accuracy throughout learning steps, shown in Figure 4 *left*. While other methods suffer from various degrees of forgetting, CLUE maintains comparatively stable even when the domain distribution in the current task changes. Figure 4 *right* shows the eventual accuracy of all learned tasks when complete training is finished. In each benchmark, the histograms of the first three tasks reflect the rigidity of the method and the last histogram of task 4 mirrors plasticity. Through this graph, CLUE maintains high accuracy in former tasks while preserving plasticity for future tasks, almost always being the closest to the upper bound accuracy.

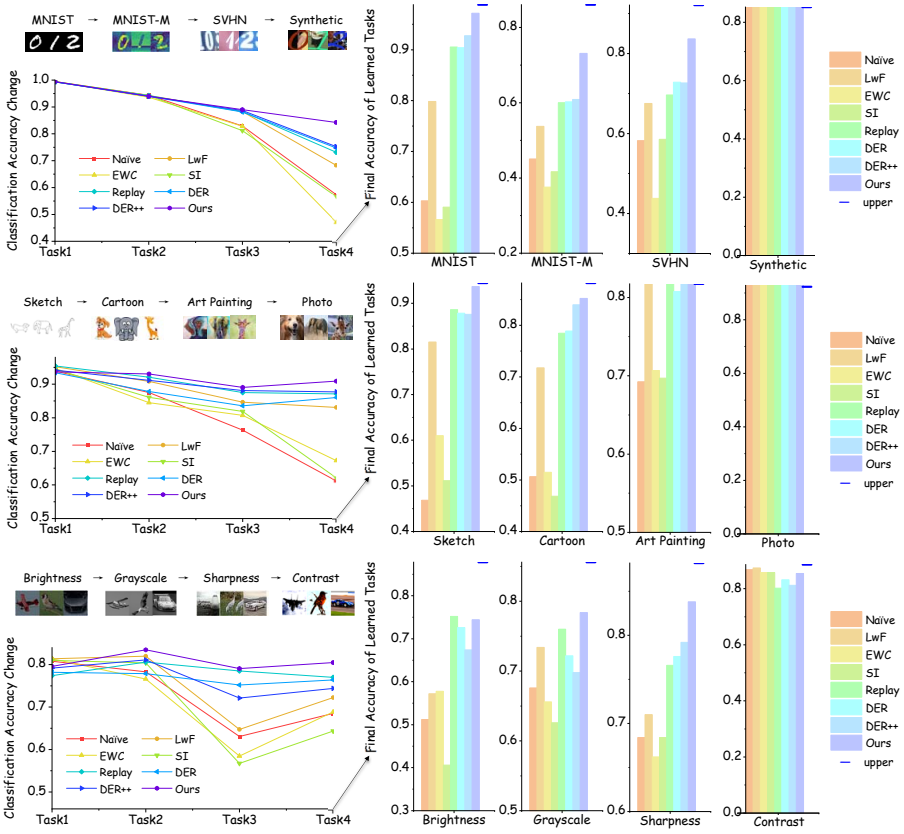


**Fig. 4.** Performance comparisons of different methods. The dynamic changes of overall accuracy are depicted in the *left* while the final accuracy distribution of all tasks after completing all tasks is in the *right*. Our method maintains high accuracy in former tasks while preserving plasticity for future tasks.

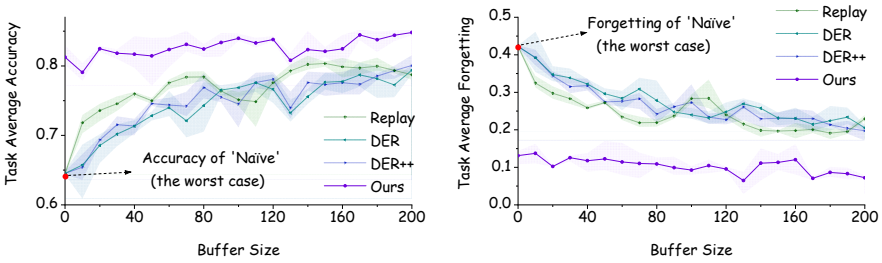


**Fig. 5.** The dynamic changes of $ACC_{avg}$ and $F_{avg}$ on the Digits benchmark using different continual learning methods when the buffer size changes from 0 to 200. The results of method 'Naive' is marked in red as a comparison. Our method is observed to be robust, facing reduced buffer sizes.

### 4.2 Buffer Size Analysis

In continual learning settings, buffer sizes (the number of replay images allowed to be stored) are fixed. Therefore, as the task number grows, stored image numbers of each task will gradually decrease, significantly impacting the performance. Superior continual learning methods must maintain high performance even when buffer sizes are limited. Thus, it is fundamental to study the changes in metrics when the buffer size becomes smaller.

In Figure 5, the dynamic changes of $ACC_{avg}$ and $F_{avg}$ is plotted for all the replay-based methods. It is observed that when the buffer size is reduced to 0, other replay-based methods degenerate into 'Naive', while CLUE still maintains decent performance. When the buffer size is gradually reduced to one-tenth, in CLUE, both $ACC_{avg}$ and $F_{avg}$ have a significantly smaller change than other methods. $ACC_{avg}$ is maintained at around 80%, while $F_{avg}$ is approximately 10%.

Therefore, we can safely conclude that CLUE is a robust method facing reduced buffer size. On the one hand, it helps maintain continual learning performance as the task number increases. On the other hand, when the task number remains unchanged, it significantly reduces the number of required replay samples, saving storage overhead.

### 4.3 Model Interpretability Analysis

As is illustrated above in Figure 2, the changes in domain distribution may lead to a distorted attention map and invalid logits outputs. In this part, we conduct experiments to interpret the effectiveness of CLUE in correcting the attention map and preserving the logits distribution.

Figure 6 shows the output logits distribution of nine typical samples from the first three tasks of the three benchmarks. Before forgetting shows the results right after learning the task to which the samples belong, while after forgetting

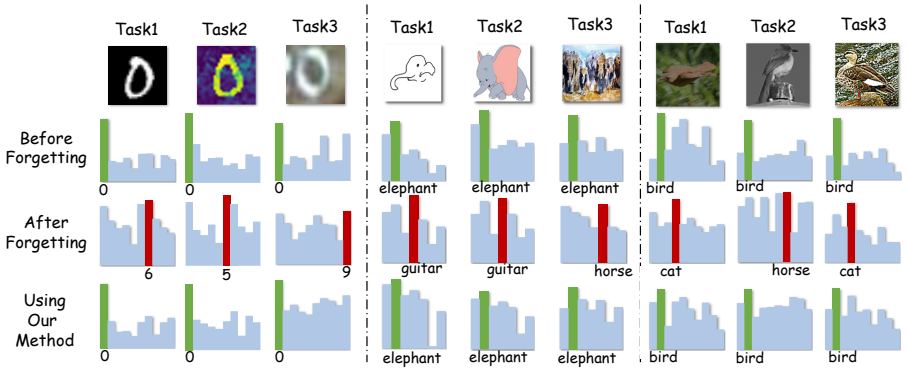


**Fig. 6.** Logits distributions of typical samples. *Left* shows the Digits benchmark, *middle* depicts the Pictures benchmark, and *right* corresponds to the Processing benchmark. CLUE works in maintaining the original logits distributions and classifying correctly.

shows the performance of the eventual model. The last row in Figure 6 shows the final results when applying CLUE.

It is observed that without our model, '0' might lose the logits distribution of the original domain, leading to the misclassification as '6', '5', or '9'. The sample 'elephant' in the Picture benchmark and 'bird' in the Processing benchmark might also be misclassified without an effective continual learning method. With CLUE, accurate classification can be achieved by maintaining the original logits distribution as much as possible to prevent catastrophic forgetting.

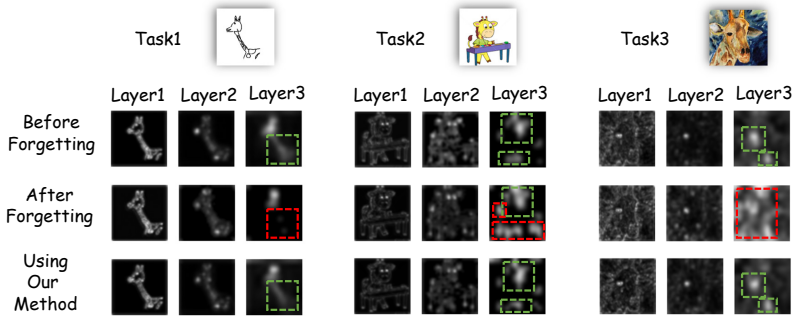


**Fig. 7.** Attention Maps of typical samples from the Picture benchmark without or with CLUE. The mean output of channels in layer3 (ResNet-18) is used as the attention map, where bright color corresponds to the vital area. CLUE works in maintaining proper attention maps and classifying correctly.

The attention maps of typical samples are shown in Figure 7. The Pictures benchmark is used as an example as its 224 * 224-sized images are large enough to be observed and analyzed.

As is shown, in the sketch image, for correct classification, the neck of the giraffe should be focused. By applying CLUE, the attribute can be maintained. The cartoon giraffe can be classified successfully by its head and legs. However, after forgetting, the neural network tends to pay more attention to table legs and the giraffe's tail. Our method can preserve important areas without being disturbed by noisy information. The eye and noise in the art painting of giraffes are the prime areas for classification. CLUE keeps them well. Therefore, applying CLUE guarantees a more accurate attention map.

### 4.4 Ablation Studies

Table 2 shows the ablation studies, where the significance of $L_{net}$ and $L_{data}$ is tested. Metrics of the classifier when removing $L_{net}$ or $L_{data}$ or both are calculated and recorded. Additionally, we test the performance where $L_{data}$ is replaced by commonly used cross-aentropy $L_{CE}$.

As is observed in Table 2, when both $L_{net}$ and $L_{data}$ are omitted, the method degrades into naive finetuning. Attaching either $L_{net}$ or $L_{data}$ will promote $ACC_{avg}$ by a large margin in the Digits ($\approx 15\%$) and Pictures ($\approx 10\%$) benchmark. In the Processing benchmark, $L_{net}$ is more effective than $L_{data}$. Regarding $F_{avg}$, $L_{net}$ works better in alliviating forgetting in all benchmarks. Obviously, utilizing both loss functions will further improve the performance.

**Table 2.** Ablation Studies

| | $\mathbf{L_{network}}$ | $\mathbf{L_{data}}$ | $\mathbf{ACC_{avg}}(\%)\uparrow$ | $\mathbf{F_{avg}}(\%)\downarrow$ |
|---|---|---|---|---|
| Benchmark 1: Digits | ✗ | ✗ | 64.51 | 42.23 |
| | ✔ | ✗ | 80.15 | 14.38 |
| | ✗ | ✔ | 79.00 | 20.96 |
| | ✔ | $\mathbf{L_{CE}}$ | 82.41 | 11.64 |
| | ✔ | ✔ | **84.81** | **7.23** |
| Benchmark 2: Pictures | ✗ | ✗ | 66.91 | 38.24 |
| | ✔ | ✗ | 87.99 | 3.78 |
| | ✗ | ✔ | 88.27 | 6.03 |
| | ✔ | $\mathbf{L_{CE}}$ | 89.12 | 2.45 |
| | ✔ | ✔ | **90.16** | **1.34** |
| Benchmark 3: Processing | ✗ | ✗ | 68.88 | 23.29 |
| | ✔ | ✗ | 76.55 | 8.27 |
| | ✗ | ✔ | 72.60 | 14.03 |
| | ✔ | $\mathbf{L_{CE}}$ | 74.25 | 7.67 |
| | ✔ | ✔ | **79.13** | **5.03** |

Besides, when changing $L_{net}$ into $L_{CE}$, drops are witnessed in the performance, with approximately 2%, 1% and 5% decrease in the $ACC_{avg}$ on the three benchmarks. It is probably because $L_{CE}$ focuses more on the between-class difference of samples, while $L_{net}$ focuses more on the class logits distributions facing new domains.

As a result, both $L_{net}$ and $L_{data}$ are effective and contributing parts for our CLUE.

## 5 Conclusion

In this paper, focused on the DI setting, we clarify the problem setting and define the metrics. Firstly, the main culprit of forgetting when training on shifted domains is found through preliminary experiments. It is because the feature extractor has adjusted to the new domain distribution, being invalid for former tasks. Next, to deal with this issue, a Consolidating Learned and Undergoing Experience (CLUE) method is proposed to mitigate forgetting in DI classification. It consists of two more loss functions to control network changes and can be applied in any mainstream classification model without introducing extra parameters or processing steps. Comprehensive experiments show the superior performance of CLUE compared with other baselines on three benchmarks - Digits, Pictures, and Processing, with higher task-average accuracy and less forgetting. Besides, extensive experiments show that CLUE remains robust when there are limited replay samples, maintaining higher performance and saving storage overhead. Moreover, both theoretical derivation and model interpretability visualization justify the feasibility of CLUE.

**Acknowledgements** This work was supported by the National Key R&D Program of China (No.2020AAA0105500) the National Natural Science Foundation of China (No. U21B2013, No.61971260)

## References

1. Biesialska, M., Biesialska, K., Costa-Jussa, M.R.: Continual lifelong learning in natural language processing: A survey. arXiv preprint arXiv:2012.09823 (2020)
2. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems **33**, 15920–15930 (2020)
3. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: Proceedings of the European conference on computer vision (ECCV). pp. 233–248 (2018)
4. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 532–547 (2018)
5. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 215–223. JMLR Workshop and Conference Proceedings (2011)

6. Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
7. Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
8. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A.A., Pritzel, A., Wierstra, D.: Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint arXiv:1701.08734 (2017)
9. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The journal of machine learning research **17**(1), 2096–2030 (2016)
10. Hayes, T.L., Kafle, K., Shrestha, R., Acharya, M., Kanan, C.: Remind your neural network to prevent catastrophic forgetting. In: European Conference on Computer Vision. pp. 466–483. Springer (2020)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learningfor image recognition. ComputerScience (2015)
12. Hinton, G., Vinyals, O., Dean, J., et al.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 **2**(7) (2015)
13. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
14. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
15. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
16. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: Proceedings of the IEEE international conference on computer vision. pp. 5542–5550 (2017)
17. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
18. Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T.L., De Lange, M., Masana, M., Pomponi, J., Van de Ven, G.M., et al.: Avalanche: an end-to-end library for continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3600–3610 (2021)
19. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. Advances in neural information processing systems **30** (2017)
20. Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., Sanner, S.: Online continual learning in image classification: An empirical survey. Neurocomputing **469**, 28–51 (2022)
21. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 67–82 (2018)
22. Mallya, A., Lazebnik, S.: 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)
23. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277 (2020)

24. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
25. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019)
26. Patel, V.M., Gopalan, R., Li, R., Chellappa, R.: Visual domain adaptation: A survey of recent advances. IEEE signal processing magazine **32**(3), 53–69 (2015)
27. Prabhu, A., Torr, P.H., Dokania, P.K.: Gdumb: A simple approach that questions our progress in continual learning. In: European conference on computer vision. pp. 524–540. Springer (2020)
28. Pytorch, A.D.I.: Pytorch (2018)
29. Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., Tesauro, G.: Learning to learn without forgetting by maximizing transfer and minimizing interference. arXiv preprint arXiv:1810.11910 (2018)
30. Roy, P., Ghosh, S., Bhattacharya, S., Pal, U.: Effects of degradations on deep neural network architectures. arXiv preprint arXiv:1807.10108 (2018)
31. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016)
32. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International Conference on Machine Learning. pp. 4548–4557. PMLR (2018)
33. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. Advances in neural information processing systems **30** (2017)
34. Simon, C., Faraki, M., Tsai, Y.H., Yu, X., Schulter, S., Suh, Y., Harandi, M., Chandraker, M.: On generalizing beyond domains in cross-domain continual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9265–9274 (2022)
35. Tang, S., Su, P., Chen, D., Ouyang, W.: Gradient regularized contrastive learning for continual domain adaptation (2021)
36. van de Ven, G.M., Siegelmann, H.T., Tolias, A.S.: Brain-inspired replay for continual learning with artificial neural networks. Nature communications **11**(1), 1–14 (2020)
37. Van de Ven, G.M., Tolias, A.S.: Three scenarios for continual learning. arXiv preprint arXiv:1904.07734 (2019)
38. Volpi, R., Larlus, D., Rogez, G.: Continual adaptation of visual representations via domain randomization and meta-learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4443–4453 (2021)
39. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018)
40. Xiang, Y., Fu, Y., Ji, P., Huang, H.: Incremental learning using conditional adversarial networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6619–6628 (2019)
41. Xie, J., Yan, S., He, X.: General incremental learning with domain-aware categorical representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14351–14360 (2022)
42. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017)
43. Zeng, G., Chen, Y., Cui, B., Yu, S.: Continual learning of context-dependent processing in neural networks. Nature Machine Intelligence **1**(8), 364–372 (2019)
44. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: International Conference on Machine Learning. pp. 3987–3995. PMLR (2017)

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