International Journal of Computer Vision (2023) 131:1825–1839
https://doi.org/10.1007/s11263-023-01775-0

Trust-Region Adaptive Frequency for Online Continual Learning

Yajing Kong[1] · Liu Liu[1] · Maoying Qiao[2] · Zhen Wang[1] · Dacheng Tao[1]

Received: 12 August 2022 / Accepted: 9 February 2023 / Published online: 18 April 2023

Abstract
In the paradigm of online continual learning, one neural network is exposed to a sequence of tasks, where the data arrive in an online fashion and previously seen data are not accessible. Such online fashion causes insufficient learning and severe forgetting on past tasks issues, preventing a good stability-plasticity trade-off, where ideally the network is expected to have high plasticity to adapt to new tasks well and have the stability to prevent forgetting on old tasks simultaneously. To solve these issues, we propose a trust-region adaptive frequency approach, which alternates between standard-process and intra-process updates. Specifically, the standard-process replays data stored in a coreset and interleaves the data with current data, and the intra-process updates the network parameters based on the coreset. Furthermore, to improve the unsatisfactory performance stemming from online fashion, the frequency of the intra-process is adjusted based on a trust region, which is measured by the confidence score of current data. During the intra-process, we distill the dark knowledge to retain useful learned knowledge. Moreover, to store more representative data in the coreset, a confidence-based coreset selection is presented in an online manner. The experimental results on standard benchmarks show that the proposed method significantly outperforms state-of-art continual learning algorithms.

Keywords Online continual learning · Catastrophic forgetting · Trust-region · Deep learning

1 Introduction

Continual learning (CL) is a learning paradigm that aims to mimic the human abilities of adapting to new environments while not forgetting past experience (Delange et al., 2021; Peng et al., 2021; Kong et al., 2022; Wang et al., 2022b, c, d). However, when the network is exposed to a sequence of tasks sequentially, the performance of the old tasks would drop significantly, which is referred to as catastrophic forgetting (McCloskey &Cohen, 1989; Ratcliff, 1990), a well-known challenge in continual learning. The problem closely involves the stability-plasticity dilemma (Grossberg, 1982; Mermillod et al., 2013), where with limited resources, the network is infeasible to have plasticity to learn a new task well and stability to maintain useful knowledge learned from past tasks simultaneously.

To alleviate the stability-plasticity dilemma, three classes of continual learning approaches are proposed: replay-based methods, architecture-based methods, and regularization-based methods. In particular, replay-based methods store historical data in a limited coreset and replay the data alongside new data (Rolnick et al., 2019; Isele &Cosgun, 2018; Chaudhry et al., 2019; Lopez-Paz &Ranzato, 2017), as done by Experience Replay (ER) (Riemer et al., 2018) and Dark Experience Replay (DER) (Buzzega et al., 2020). Architecture-based methods augment the networks or allocate subnetworks for new tasks to decrease the interference on past works (Zhou et al., 2012; Jerfel et al., 2019; Mallya

Communicated by Diane Larlus.

This work is supported by Australian Research Council Project FL-170100117.

✉ Dacheng Tao
dacheng.tao@gmail.com

Yajing Kong
ykon9947@sydney.edu.au

Liu Liu
liu.liu1@sydney.edu.au

Maoying Qiao
maoying.qiao@uts.edu.au

Zhen Wang
zwan4121@sydney.edu.au

[1] School of Computer Science, Faculty of Engineering, University of Sydney, Sydney NSW, 2008, Australia

[2] University of Technology Sydney, Sydney NSW,, Australia

&Lazebnik, 2018), e.g., Progressive Neural Networks (PNN) (Rusu et al., 2016). Regularization-based methods prevent the variants of important parameters by adding a penalty term to the loss function (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018; Lee et al., 2017), like Elastic Weight Consolidation (EWC) (Schwarz et al., 2018).

However, most of the approaches are tailed for offline setting, where the model could iterate the entire dataset of each task multiple times. That is, the model can access the whole dataset of the current task at a time and needs additional storage to store the whole current dataset, which is not realistic. Therefore, we consider a more restrictive but practical setting, online continual learning (online CL). Specifically, online CL requires continual learning algorithms to observe the data of each task in a single pass while previous data are unavailable.

Considering the excellent performance of replay-based methods in continual learning, in this paper, we devote to replay-based method that stores a subset of historical data in a coreset. The typical problem of replay-based methods is data imbalance where the data of the current task and the data of previous tasks are imbalanced due to the inaccessibility of the old data and the small size of coreset. Moreover, when applied to online fashion, replay-based methods further face more challenges, preventing them from achieving a good stability-plasticity trade-off. For example, the model only observes the data stream from sequential tasks in a single pass, resulting in unsatisfactory learning of tasks (poor plasticity) and severe catastrophic forgetting (poor stability). Moreover, for online CL, the typical sampling strategy, *reservoir* randomly samples a uniform subset from the input stream and would omit the representative and informative data of old tasks, resulting in more forgetting of previous tasks.

Therefore, to overcome the challenges, we propose a new online continual learning approach, Trust-Region Adaptive Frequency (TRAF), which alternates between standard-process and intra-process updates based on a trust region. Specifically, the standard-process trains from data stored in a coreset and interleaves the data with current data, and the intra-process updates the network parameters based on the coreset. By triggering the intra-process during the standard-process, the model could improve the performance of tasks stemming from insufficient learning and alleviate the forgetting on previous tasks simultaneously. Moreover, to alleviate the data imbalance, intra-process is better to be triggered more frequently in the stage that the coreset is more balanced. We propose a trust-region inspired approach (Nocedal and Wright, 2006; Conn et al., 2000; Cartis et al., 2011), measured by confidence score, to detect the stage and adjust the frequency of the intra-process based on the trust region. During intra-process, we further distill the dark knowledge to retain learned knowledge. Finally, considering the importance of storing data, we introduce a confidence-based coreset selection to store more representative samples to further alleviate the forgetting. The full procedure of the proposed method is shown in Fig. 1.

The experimental results on different benchmarks demonstrate that TRAF could outperform existing competitive continual learning algorithms by a considerable margin. To summarize, our contributions are threefold:

- For online CL, we propose a new online CL method, Trust-Region Adaptive Frequency (TRAF), which alternates between standard-process and intra-process updates based on a trust region, to relieve the stability-plasticity dilemma.
- To further improve performance, TRAF also uses confidence-based coreset selection to select more representative data.
- Extensive experimental results on two standard protocols and several standard benchmarks show that the proposed method could achieve state-of-art performance.

2 Related Works

2.1 Continual Learning Approaches

Replay-based methods are a prominent class of continual learning approaches and achieve state-of-the-art performance in many challenging scenarios. Specifically, replay-based methods maintain a small memory buffer to store data and train the historical data interleaved with the new data at the latter training iterations (Rolnick et al., 2019; Isele &Cosgun, 2018; Chaudhry et al., 2019; Shin et al., 2017; Rao et al., 2019; Aljundi et al., 2019a, 2017; Hou et al., 2018; Ostapenko et al., 2019; Bang et al., 2021). For instance, Experience Replay (ER) is the most classical approach that jointly optimizes the model parameters by replaying the old data alongside new data. Gradient Episodic Memory (GEM) (Lopez-Paz &Ranzato, 2017) and Averaged-GEM (AGEM) (Chaudhry et al., 2018b) update the model under inequality constraints of gradients, which is computed by the gradients of the stored samples. Incremental Classifier and Representation Learning (iCaRL) (Rebuffi et al., 2017) learns in a class-incremental way by storing samples that are close to the center of each class. Rethinking-Experience Replay (RE-ER) (Buzzega et al., 2021) proposes several simple techniques to tackle the existing challenges in ER. Gradient based Sample Selection (GSS) (Aljundi et al., 2019b) focuses on the selection strategy and proposes a variation on ER from the view of constrained optimization. Meta-Experience Replay (MER) (Riemer et al., 2019) combines ER with optimization-based meta-learning to maximize transfer from the past tasks while minimizing interference. DER++ (Buzzega et al., 2020) promotes consistency with the past by matching the

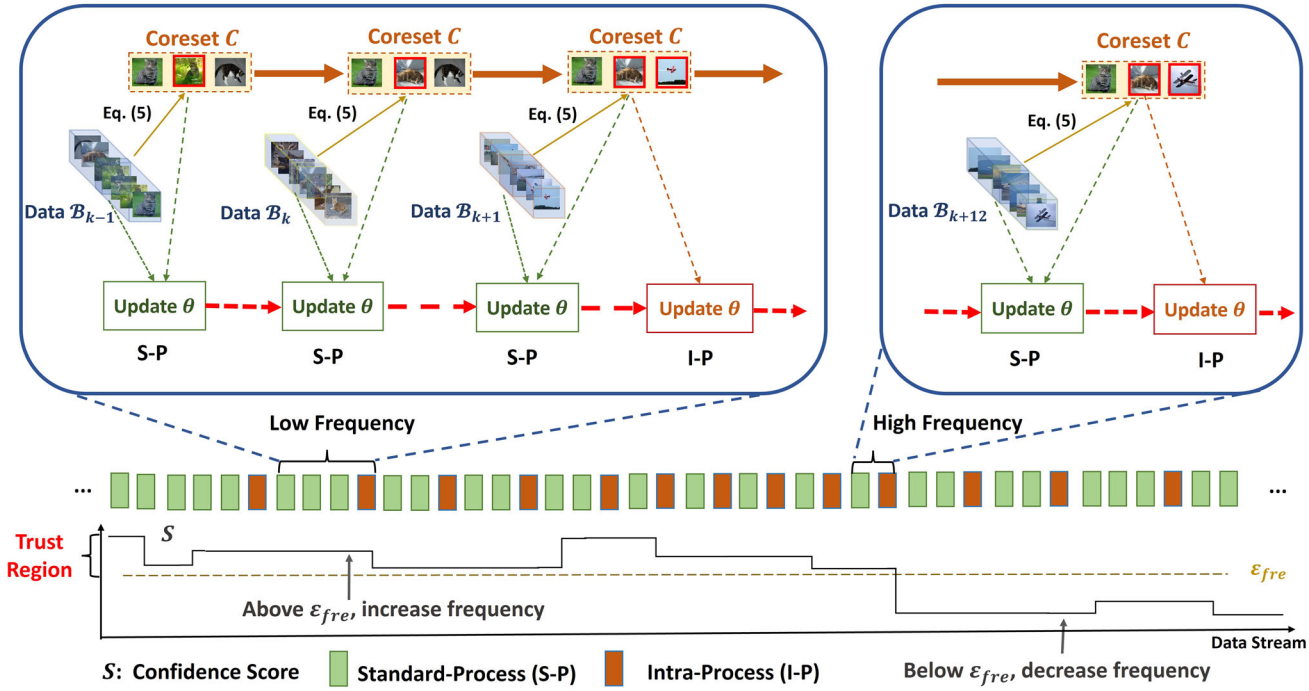


Fig. 1 The full procedure of the proposed method. The procedure includes Standard-Process (S-P) and Intra-Process (I-P), where S-P updates the model based on the current data and coreset, and I-P updates based on coreset. The frequency of I-P is dependent on the confidence score S. When the average of the confidence score is higher than the threshold ϵ_{fre}, the frequency is gradually increased. Otherwise, we decrease the frequency. *Low frequency* is referred to the situation where the number of standard-process is much larger than the I-P within a certain time and vice versa. The changes of the confidence score are only used for illustration

network's outputs selected throughout the optimization trajectory. Unsupervised Continual Learning (UCL) (Madaan et al., 2022) mixes up new examples with past examples to mitigate the forgetting. Among the mentioned methods, all methods are applicable or easily modified to the setting of online CL, except for iCaRL and GEM.

Architecture-based methods expand the network progressively when needed or allocate different parameters for different tasks (Serra et al., 2018; Mallya &Lazebnik, 2018; Mallya et al., 2018; Li et al., 2019; Zhou et al., 2012; Wu et al., 2020; Yoon et al., 2019). For example, PNN (Rusu et al., 2016) expands the networks when the new task comes, and retain the networks learned on past tasks. Wu et al. (2020) progressively and dynamically grows neural networks by jointly optimizing the network. However, these methods may result in a cumbersome and complex model if new tasks continually arrive.

Regularization-based methods add a penalty term to the loss function to prevent the changes of network parameters (Chaudhry et al., 2018a; Yin et al., 2020; Nguyen et al., 2017; Ritter et al., 2018; Lin et al., 2022). For example, EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), MAS (Aljundi et al., 2018), and ALASSO (Park et al., 2019) are devoted to the computation of parameters' importance while LwF (Li and Hoiem, 2017) aims to distill the knowledge without storing old data. However, these approaches may lead to unsatisfactory performance without access to previous data, especially in challenging scenarios.

2.2 Online Continual Learning

While the majority of continual learning methods are designed for unsuitable scenarios, where the model can iterate on the entire dataset of the current task multiple times (Zenke et al., 2017; Schwarz et al., 2018; Rusu et al., 2016; Rebuffi et al., 2017), online continual learning (online CL) has been gaining much interest recently due to its ubiquitous in many real-world problems. In this paper, we consider a challenging task that is more restrictive, i.e., online CL (Jin et al., 2021; Sun et al., 2022; Aljundi et al., 2019b). In online CL, the model observes the data of each task in a single pass and previous data are unavailable.

Moreover, recent works (Delange et al., 2021; Buzzega et al., 2020) also provided the requirements that the continual learning methods should focus on to be more applicable in practical: (a) *no task boundaries*: the model does not rely on the task boundaries. (b) *constant memory*: the memory is bounded throughout the entire training phase. (c) *no test time oracle*: the task identities which are used to select the relevant task for each image are not accessible at inference time. Our

Fig. 2 The illustration of online data stream where the data of each task arrives sequentially and each data can be only observed once

setting follows the guidelines and according to the fact that whether the task identities (time oracle) or not, we divide the scenario into two protocols: *Task-Aware* (with task identities) and *Task-Free* (without task identities) (Pham et al., 2021), and evaluate the proposed method on both protocols.

3 Problem Setting

In this section, we present the setting of online continual learning. Figure 2 shows the illustration of online CL. Formally, in online CL, the model is learned on a sequence of image classification tasks $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$, where T is the total number of tasks and $\mathcal{T}$ is the tasks set. For task $\mathcal{T}_t$, the input samples x and the corresponding labels y are drawn from the independently and identically distributed distribution of task $\mathcal{T}_t$. Let $\mathcal{D}_t$ be the dataset of task $\mathcal{T}_t$, and $\mathcal{D}$ be the corresponding online data stream consisting of all datasets $\mathcal{D}_t, t \in \{1, 2, \ldots, T\}$, sequentially. Note that the task boundaries are not provided to indicate the coming of a new task during training. The model is trained on a sequence of batches $\{\mathcal{B}_1, \mathcal{B}_2, ...\}$ from $\mathcal{D}$ with each data seen once.

Let θ be the model parameters and $\mathcal{N}$ be the network. In this paper, we focus on replay-based methods, a prominent class of approaches in continual learning, which store a subset of past data in a limited replay coreset $\mathcal{C}$ and replay the data in the future (Buzzega et al., 2020; Madaan et al., 2022; Buzzega et al., 2021). $|\mathcal{A}|$ denotes the datasize of $\mathcal{A}$.

4 Methodology

In this section, we depict the proposed method, Trust-Region Adaptive Frequency (TRAF), which alternates between standard-process and intra-process updates in an adaptive frequency. We first describe the standard-process and intra-process (Sect. 4.1) and then introduce the trust-region adaptive frequency for intra-process (Sect. 4.2), the key idea of our work. To further alleviate the catastrophic forgetting, we also propose confidence-based coreset selection to select more representative data (Sect. 4.3). Finally, we discuss the difference between our work and some related works (Sect. 4.4).

Algorithm 1 Trust-Region Adaptive Frequency in Online Continual Learning

Input: Network $\mathcal{N}$, Parameters θ, Data stream $\mathcal{D}$, Learning rate η, Scalars m, λ, δ, ϵ_{fre}, $inv_{\min}$, $inv_{\max}$, ϵ_{ccs}, inv_0 and β,
Output: Target network $\mathcal{N}$

$k = 1, \mathcal{C} = \{\}$
for $\mathcal{B}$ in $\mathcal{D}$ **do**
  $\hat{Y}_{\mathcal{B}} = \mathcal{N}(\mathcal{B})$
  $inv_k' = \begin{cases} inv_{k-1}' - \delta, & \text{avg}(\mathcal{S}(\mathcal{B};\theta)) \geq \epsilon_{\text{fre}}, \\ inv_{k-1}' + \delta, & \text{otherwise} \end{cases}$
  $inv_k = \begin{cases} \max\{\lfloor inv_k' \rfloor, inv_{\min}\}, & \text{avg}(\mathcal{S}(\mathcal{B};\theta)) \geq \epsilon_{\text{fre}} \\ \min\{\lceil inv_k' \rceil, inv_{\max}\}, & \text{otherwise} \end{cases}$
  if $\mathbb{I}(k, inv_k) = 1$ **then**
    $\theta \leftarrow \theta - \eta\nabla_\theta(\mathcal{L}(\mathcal{C};\theta) + \lambda\mathcal{L}_d(\tilde{Y}_{\mathcal{C}}, \hat{Y}_{\mathcal{C}}))$
  end if
  $\theta \leftarrow \theta - \eta\nabla_\theta(\mathcal{L}(\mathcal{B};\theta) + \beta\mathcal{L}(\mathcal{C};\theta))$
  $\textbf{idx}^* = \begin{cases} \underset{n\in[|\mathcal{B}|]}{\text{argmax}^{(m)}}\ \mathcal{S}(\mathcal{B};\theta), & \text{avg}(\mathcal{S}(\mathcal{B};\theta)) \geq \epsilon_{\text{ccs}}, \\ random(m, |\mathcal{B}|), & \text{otherwise} \end{cases}$
  $\mathcal{C} \leftarrow reservoir(\mathcal{C}, \mathcal{B}[\textbf{idx}^*], \hat{Y}_{\mathcal{B}}[\textbf{idx}^*])$ ▷ Alg. 2
  $k = k + 1$
end for
return $\mathcal{N}$

4.1 Standard-Process and Intra-Process

4.1.1 Standard-Process

In this subsection, we first introduce the experience replay (ER), the most typical replay-based method, which stores a subset of historical data across encountered tasks and optimizes the network with the historical data and current data during training. Formally, when training on the current batch $\mathcal{B}$, the objective can be represented as the following:

$$\mathcal{L}(\mathcal{B};\theta) + \beta\mathcal{L}(\mathcal{C};\theta), \tag{1}$$

where $\mathcal{L}$ is cross entropy loss, $\mathcal{C}$ is the coreset containing the stored training samples, θ are the model parameters, and β is a factor that controls the balance between the new task and

past tasks. We call the updates in Eq. (1) as *Standard-Process* (S-P).

ER and its variants (Buzzega et al., 2020; Arani et al., 2022; Buzzega et al., 2021) have achieved impressive achievements in conventional continual learning. However, they are still hard to achieve satisfactory stability-plasticity trade-off on the online CL. Specifically, the model observes batch $\mathcal{B}$ sequentially, where each batch is seen once, resulting in insufficient learning of the current task (poor plasticity). Moreover, except for the historical data stored in the coreset, the model can only access the data of the current task, leading to more attention to the classes in the current task. The phenomenon would result in more severe catastrophic forgetting of previous classes (poor stability).

4.1.2 Intra-Process

Inspired by previous works that multiple iterations on the data can improve the unsatisfactory performance (Tang et al., 2021), to maximally utilize the limited data in the coreset, we introduce a new process, *Intra-Process* (I-P), updating the model parameters under the coreset. Formally, the loss function for the intra-process is $\mathcal{L}(\mathcal{C};\theta)$, where $\mathcal{L}$ is the cross entropy loss, $\mathcal{C}$ is the coreset containing the stored training samples.

To improve the insufficient learning of tasks, we alternate the standard-process with intra-process throughout the optimization trajectory. Specifically, we trigger the intra-process at a certain frequency during the training of the standard-process, where the frequency corresponds to the triggers of intra-process in certain iterations. We define the trigger function as

$$\mathbb{I}(k, inv) = \begin{cases} 1, & k \mod inv = 0, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where *mod* is the operation of modulo, k is the current iteration number and *inv* is an integer that controls the frequency and it is negatively related to the frequency, i.e., larger *inv* corresponds to lower frequency.

4.2 Trust-Region Adaptive Frequency for Intra-Process

If intra-process updates parameters based on the more balanced coreset, it can alleviate the negative impact brought by data imbalance. However, due to the online setting, the class balance in the coreset varies at the different learning stages of the current task. As found in our experiments and shown in Fig. 3, the class distribution in coreset is more uniform for all observed classes at the late stage of the current task learning. Therefore, intra-process should be triggered more frequently at the late stage of the current task to alleviate the negative impact brought by data imbalance better. However, in online learning, the boundaries of the task are not accessible thus we could not obtain the learning stage of the current task. To this end, we designed an approach to detect the late stage of the current task. We find that, due to online fashion that data can be seen only once, the performance of the model on the current task would be better at the later training stage of the current task. Therefore, we use the performance of the current task to detect the latter stage of the current task. A natural way to measure performance is the confidence score, where a higher confidence score represents better performance. Therefore, to detect the stage, we propose a trust-region that is measured by a confidence score and

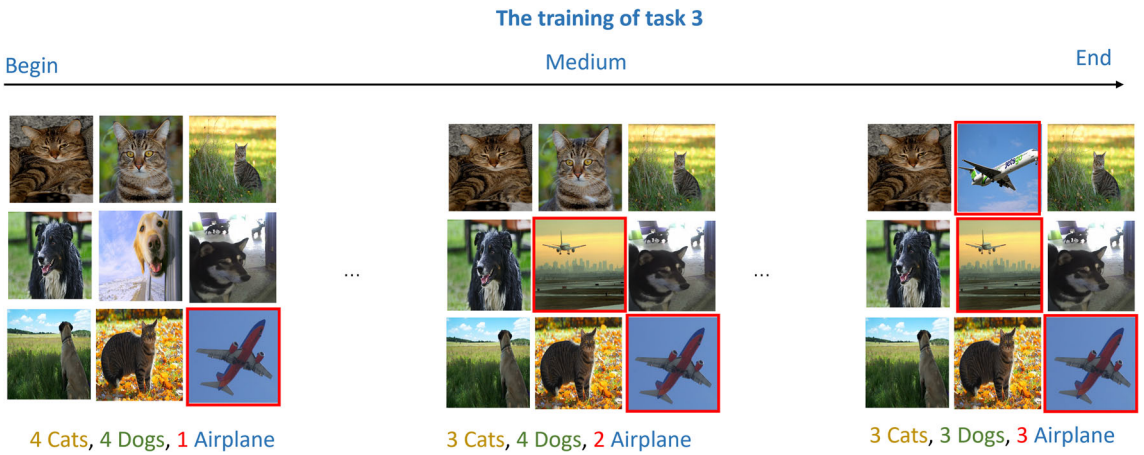


Fig. 3 The illustration of distribution changes in the coreset during the training of a task. Assume that the class of the current task is Airplane. The samples in the coreset are all from old tasks at the beginning of current task learning and then the fraction of the samples from the current task will gradually increase until the end of the task. Finally, the samples of each class will be almost equal

adjust the frequency dynamically based on the trust region, called trust-region adaptive frequency.

Specifically, to represent the region explicitly, we use the average confidence score, which is the predicted probability of the ground truth label, for the current batch to measure the performance of the current task. When the model is under the trust region, we increase the frequency by decreasing the factor inv and we decrease the frequency if the model is outside the region. Higher score is trusted because it represents better performance, a later stage of current task learning, and a more balanced coreset. Therefore, let $\mathcal{S}(x;\theta)$ be the confidence score of x. Then the candidate of inv_k can be updated by

$$inv_k^{'} = \begin{cases} inv_{k-1}^{'} - \delta, & \text{avg}(\mathcal{S}(\mathcal{B};\theta)) \geq \epsilon_{\text{fre}}, \\ inv_{k-1}^{'} + \delta, & \text{otherwise}, \end{cases} \tag{3}$$

where avg$(\cdot)$ denotes the average function, ϵ_{fre} is the threshold of the score avg$(\mathcal{S}(\mathcal{B};\theta))$, k is the current iteration number, δ is the amplitude of the frequency update and $\mathcal{B}$ is the current batch. After obtaining the $inv_k^{'}$, we round up or round down it to obtain the inv_k used in the trigger function:

$$inv_k = \begin{cases} \max\{\lfloor inv_k^{'} \rfloor, inv_{\min}\}, & \text{avg}(\mathcal{S}(\mathcal{B};\theta)) \geq \epsilon_{\text{fre}} \\ \min\{\lceil inv_k^{'} \rceil, inv_{\max}\}, & \text{otherwise}, \end{cases} \tag{4}$$

where $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the operations of rounding down and rounding up, respectively; $inv_{\max}$ and $inv_{\min}$ are the maximum and minimum value of inv_k, respectively.

As shown in Fig. 1, when the average confidence score of the current batch is satisfactory, i.e., higher than the threshold ϵ_{fre}, we decrease the inv and the corresponding frequency of intra-process is increased. Otherwise, we increase inv and then the frequency is decreased. Note that ϵ_{fre} is an important factor because it determines the trust region. For example, when ϵ_{fre} is large, the performance of the current task is better, and the classes in the coreset are more balanced. However, in the situation, most of the region is in the untrust region, and the triggers of intra-process are lower through the optimization trajetory, impacting the performance of the model. Moreover, ϵ_{fre} is related to the complex of dataset. When the dataset is easy to learn, then ϵ_{fre} should be a larger value since the worse case can also be well classified.

To further relieve forgetting and maintain the useful knowledge learned from the past, we distill the dark knowledge (Buzzega et al., 2020; Gou et al., 2021; Zhao et al., 2021; Wang et al., 2020), called Dark Knowledge Distillation (DKD), during the intra-process. Specifically, we retain the network's logits and use the modified cross-entropy loss as the distillation loss. During intra-process, we sample the examplers $(x, \tilde{y}_{\mathcal{C}})$ from the coreset randomly, where $\hat{y}_{\mathcal{C}}$ is the record logits of x. Then distillation loss can be represented as:

Algorithm 2 *reservoir*$(\mathcal{C}, x, y)$

Input: coreset $\mathcal{C}$, seen examples number N, example x, label y
$C = |\mathcal{C}|$
if $C \geq N$ **then**
  $\mathcal{C}[N-1] \leftarrow (x, y)$
else
  $j = randomInteger(0, N)$
  if $j < C$ **then**
    $\mathcal{C}[j] \leftarrow (x, y)$
  end if
end if
return Coreset $\mathcal{C}$

$$\mathcal{L}_d(\tilde{y}_{\mathcal{C}}, \hat{y}_{\mathcal{C}}) = -\sum_{l=1}^{L} \tilde{y}_{\mathcal{C}}^{\prime(l)} \log \hat{y}_{\mathcal{C}}^{\prime(l)}, \tag{5}$$

where $\tilde{y}_{\mathcal{C}}^{\prime(l)} = \frac{\exp(\tilde{y}_{\mathcal{C}}^{(l)}/\tau)}{\sum_i \exp(\tilde{y}_{\mathcal{C}}^{(i)})\tau}$, $\hat{y}_{\mathcal{C}}^{\prime(l)} = \frac{\exp(\hat{y}_{\mathcal{C}}^{(l)}/\tau)}{\sum_i \exp(\hat{y}_{\mathcal{C}}^{(i)}/\tau)}$, L is the total number of classes, τ is the temperature factor, and $\tilde{y}_{\mathcal{C}}$ and $\hat{y}_{\mathcal{C}}$ are the record and current logits of x.

To the end, the training procedure can be represented as

$$\text{Standard-Process:} \quad \mathcal{L}(\mathcal{B};\theta) + \beta\mathcal{L}(\mathcal{C};\theta), \tag{6}$$

$$\text{Intra-Process :} \quad \mathcal{L}(\mathcal{C};\theta) + \lambda\mathcal{L}_d(\tilde{Y}_{\mathcal{C}}, \hat{Y}_{\mathcal{C}}), \tag{7}$$

where λ is a factor that controls the importance of distillation; $\tilde{Y}_{\mathcal{C}}$ and $\hat{Y}_{\mathcal{C}}$ are the recorded and current logits of examples randomly sampled from the coreset $\mathcal{C}$, respectively; β and λ are balanced hyperparameters which are commonly used in CL (Buzzega et al., 2020). The intra-process is happened when $\mathbb{I}(k, inv) = 1$ (defined in Eq. 2). The procedure is shown in Algorithm 1.

4.3 Confidence-Based Coreset Selection

For replay-based methods, especially in online CL, a key problem is how to choose representative data that are beneficial for future rehearing. A compatible selection strategy for online CL is the *reservoir* (Vitter, 1985), which randomly smaples a uniform subset from the input stream. Specifically, *reservoir* randomly chooses $C = |\mathcal{C}|$ samples to store in the coreset $\mathcal{C}$, guaranteeing that all seen samples have the same probability $\frac{C}{N}$ of being stored in the coreset, where N is the number of seen samples participating in the *reservoir* sample strategy. The algorithm of *reservoir* is shown in Algorithm 2, where $randomInteger(\min = 0, \max = N)$ denotes the operation that randomly selects an integer between 0 and $N-1$.

However, *reservoir* puts equal importance on all samples, which does not take data representation into consideration. Therefore, we design a simple but effective sampling strategy that could store more representative data, called Confidence-based Coreset Selection (CCS), by storing data with higher

confidence scores in an online manner. The confidence-based coreset selection relies on the confidence score to select the samples. However, at the early stage of each task learning, the confidence scores are unreliable because the model does not fit well with the current task. Therefore, we only selectively choose the samples based on the confidence score when the confidence score is reliable, i.e., the average confidence score is higher than a threshold. Or we randomly select the samples to avoid negative effect the brought by the unreliable confidence score. Formally, the indexes $\mathbf{idx}^*$ of the selected data for the current batch $\mathcal{B}$ can be formulated as

$$\mathbf{idx}^* = \begin{cases} \underset{n}{\text{argmax}}^{(m)}\, \mathcal{S}(\mathcal{B};\theta), n \in [|\mathcal{B}|], & \text{avg}(\mathcal{S}(\mathcal{B};\theta)) \geq \epsilon_{\text{ccs}}, \\ random(m, |\mathcal{B}|), & \text{otherwise}, \end{cases} \tag{8}$$

where $\underset{n}{\text{argmax}}^{(m)}$ select m indexes of the examples with top-m confidence scores from $n \in \{1, 2, \ldots, |\mathcal{B}|\}$, $\mathcal{B}$ is the current batch, $m = p \times |\mathcal{B}|$, $p \in (0, 1]$ is the ratio of selected indexes, and $random(m, |\mathcal{B}|)$ is a function that randomly select m numbers from $\{1, 2, \ldots, |\mathcal{B}|\}$ without replacement. ϵ_{ccs} is a factor that determines when the representative samples are convincing.

Therefore, the coreset $\mathcal{C}$ can be updated as following

$$\mathcal{C} \leftarrow reservoir(\mathcal{C}, \mathcal{B}[\mathbf{idx*}], \hat{Y}_{\mathcal{B}}[\mathbf{idx*}]), \tag{9}$$

where *reservoir* denotes the operation of *reservoir* sampling, **idx*** is obtained based on Eq. (8), $\hat{Y}_{\mathcal{B}}$ are the corresponding logits of the current batch. The full algorithm is shown in Algorithm 1.

4.4 Discussion

Our work is related to Liu et al. (2021) and Hou et al. (2019). However, our method differs from theirs in many aspects. First, our method does not rely on the oracle of task boundary, i.e., knowing the end of the task, to obtain a balanced coreset. Unlike Liu et al. (2021) and Hou et al. (2019) that rely on the task boundaries to obtain the balanced coreset, our proposed method does not obtain the balanced coresets directly but uses the confidence score to detect the training stage and judge the balance of the coresets. Second, both intra-process and standard-process update the network parameters and do not use additional parameters or fix parameters. For example, Liu et al. (2021) uses additional scaling weights at a neuron level and the aggregation weights. Third, in our method, we alternate standard-process and intra-process in a dynamic frequency. However, Hou et al. (2019) applies the class balance finetuning at the end of the task (phase). Liu et al. (2021) alternates the two optimization process at each iteration.

5 Experiments

In this section, we first describe the experimental setup and implementation. Then, we evaluate the continual learning algorithms on two protocols: *Task-Aware* and *Task-Free*. We also conduct ablation studies to explore the effect of different factors and show more results.

5.1 Experimental Setup and Implementation

Settings Based on the fact that whether the task identities are provided to select the relevant classifier for each image during testing, online CL can be divided into two protocols (Pham et al., 2021): *Task-Aware* and *Task-Free*, where the latter is more challenging because the task identities are unavailable at inference time.

Benchmarks Following previous works (Buzzega et al., 2020; Madaan et al., 2022; Buzzega et al., 2021), we evaluate the algorithms on four standard continual learning benchmarks: Split MNIST (S-MNIST), Split CIFAR-10 (S-CIFAR-10), Split CIFAR-100 (S-CIFAR-100), and Split TinyImageNet (S-TinyImageNet). Split MNIST and Split CIFAR-10 split the training examples of MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009) into five tasks, respectively. Each task has two disjoint classes. Split CIFAR-100 consists of 20 tasks, each of which introduces 5 classes out of the 100 classes of CIFAR-100 (Krizhevsky et al., 2009) without replacement. TinyImageNet (Stanford, 2015) consists of 100000 64 × 64 color training samples and 10000 validation images. Similarly, Split TinyImageNet is constructed by 10 sequential tasks divided from TinyImageNet. Each task has 20 disjoint classes out of the total 200 classes without replacement.

Architectures Adhere to previous works (Buzzega et al., 2020; Mirzadeh et al., 2020; Jin et al., 2021), for Split MNIST, we employ a two-layer fully connected network, where each hidden layer has 100 ReLU units. For the variants of CIFAR-10 and CIFAR-100, we employ a lightweight ResNet-18 with three times smaller than standard ResNet-18. For Split TinyImageNet, we use the standard ResNet-18 (He et al., 2016). All tasks share the same classifier, i.e., we use a single-head setting, a more challenging setting.

Baselines We compare TRAF with the following 10 methods: ER, MER (Riemer et al., 2019), AGEM (Chaudhry et al., 2018b), GSS (Aljundi et al., 2019b), UCL (Madaan et al., 2022), DER (Buzzega et al., 2020), RE-ER (Buzzega et al., 2021), DER++ (Buzzega et al., 2020), Complementary Learning System-ER (CLS-ER) (Arani et al., 2022)[1]), and

[1] Since the performance of stable model in CLS-ER is worse for online CL, we evaluate the performance on its working model.

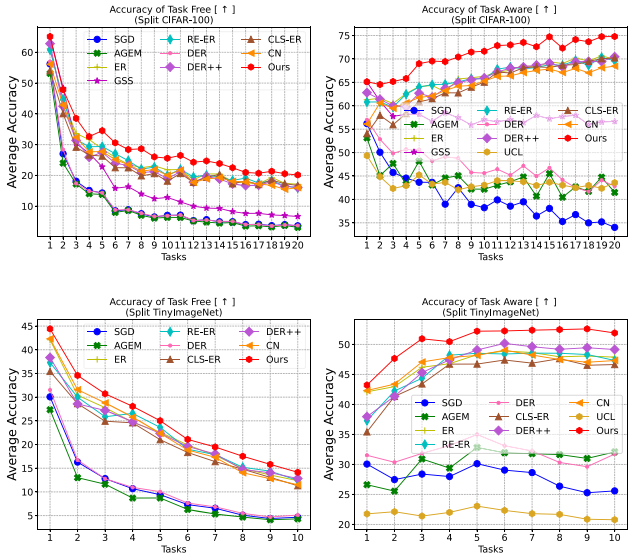


Fig. 4 The curves of the average accuracy when the network has been trained on each task on the datasets of Split CIFAR-100 and Split TinyImageNet over five runs. [↑] denotes higher is better

Continual Normalization (CN) (Pham et al., 2022). We also provide the performance of SGD (Ghadimi and Lan, 2013), which simply trains the model without any countermeasure to forgetting.

Evaluation Metric Following previous works (Mirzadeh et al., 2020; Lopez-Paz &Ranzato, 2017; Chaudhry et al., 2018b), we evaluate continual learning algorithms with two metrics: Average Accuracy (ACC) and Forgetting (FT). Formally, after the model has finished learning all tasks, ACC is the average accuracy evaluated across all observed tasks, defined as, $\text{ACC} = \frac{1}{T}\sum_{i=1}^{T} a_{Ti}$, where a_{ti} is the accuracy of the task $\mathcal{T}_i$ when the model has been learned on the task $\mathcal{T}_t$. FT measures the performance degradation of tasks from the task's peak performance to its final performance, i.e., $\text{FT} = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{t\in[T-1]}(a_{ti} - a_{Ti})$. Higher ACC and lower FT are better. With similar ACC, the algorithm with lower FT is better.

Implementation Details We use Pytorch [2] to implement the proposed algorithm and other experiments. We use the SGD optimizer and batch size of 10 for all experiments. Adhering to previous work (Buzzega et al., 2020), the coreset size of Split MNIST and Split CIFAR-10 is 200 and 500, respectively. For Split CIFAR-100 and Split TinyImageNet, the coreset size is 1000. The learning rate for all experiments is 0.03.[3] For the method-related hyperparameters of all baselines, e.g., α in DER++ and so on, we refer to the setting of the released code. CN is used on top of Experience Replay (ER). For the proposed method, the batch size for intra-process are 50 for 5-Split MNIST and 5-Split CIFAR-10, and 100 for other datasets. The sample ratio p is 0.9 for 5-Split MNIST and 5-Split CIFAR-10, and 0.8 for other datasets. We set τ to 2 for Split MNIST and 1 for other datasets. The $inv_{\max}$ for all datasets is 5, and the $inv_{\min}$ is 1 for Split CIFAR10, and 2 for other datsets. Other hyperparameters settings are shown in Table 1. The loss function is cross-entropy loss. We perform all experiments five times with different random seeds, and the results are the average results over five runs. We use *reservoir* sampling strategy (Vitter, 1985) for all baselines applicable to online CL.

5.2 Experimental Results

The effect of trust-region adaptive frequency We first assess the ACCs and FTs of constant frequency [4] and adaptive frequency on the setting of *Task-Free*, a more restrictive and challenging scenario. For a fair comparison, we exclude the component of CCS and DKD. According to Table 2, using adaptive frequency can achieve higher ACCs (relative improvement of at least 4.38%) and lower FTs than using constant frequency (any integer between the range [2, 5]), validating that using trust-region adaptive frequency could achieve better performance and less catastrophic forgetting.

Comparisons with baselines on the setting of Task-Free Table 3 summarizes the experimental results of ACCs and FTs on the protocol of *Task-Free*. According to Table 3, the proposed method could outperform baselines by a considerable margin. For example, the ACCs of TRAF are at least 1.0% higher than that of other methods on all benchmarks. Especially, on Split CIFAR-100, the ACC of TRAF achieves at least 3 % improvement over SOTA (CLS-ER). One reason for the worse performance of other methods may be that previous works are not suitable for the realistic and challenging scenarios. For instance, UCL requires multiple accesses to the datasets to learn invariant features between all tasks, while the insufficient learning stemming from the online setting prevents them from learning such representations. Therefore, when applied to the online scenario, its performance is poorer than that of our method. Moreover, Table 3 shows that the proposed method has lower or comparable forgetting with baselines, demonstrating the effectiveness of TRAF in alleviating forgetting. For example, on Split CIFAR-10, the FT of TRAF is at least 2.00 % lower than other methods. On the Split TinyImageNet, the forgetting of AGEM is lower than

[2] https://pytorch.org/

[3] UCL uses a momentum of 0.9 and weight decay of 0.0005 as the official code.

[4] constant frequency means that applying intra-process in a certain frequency.

Table 1 The hyperparameter settings

Param	S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
	Task-Free	Task-Aware	Task-Free	Task-Aware	Task-Free	Task-Aware	Task-Free	Task-Aware
β	4.5	4.5	1.0	1.0	0.5	0.5	0.5	0.5
λ	20	10	0.35	0.32	0.12	0.24	0.01	0.3
ϵ_{fre}	0.6	0.6	0.2	0.2	0.2	0.2	0.1	0.1
ϵ_{ccs}	0.7	0.7	0.175	0.175	0.2	0.2	0.1	0.1

Table 2 Comparisons of constant frequency and adaptive frequency

inv	2	3	4	5	Adaptive
ACC [↑]	18.94±1.60	16.84±1.53	17.88±1.20	17.20±1.43	**19.77**±1.71
FT [↓]	48.97±2.23	51.88±1.79	51.48±1.39	52.26±1.37	**48.86**±1.23

The bold values denote the best performance
We set the range of interval inv as [2, 5]. The dataset is the Split CIFAR-100 and the experiments are performed five runs. [↑] Higher is better. [↓] lower is better

Table 3 Results of ACC (%) [↑] and forgetting (%) [↓] evaluated on all tasks after finishing learning all tasks on the setting of *Task-Free*

Method	S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]
SGD	19.68±0.04	99.20±0.08	18.70±0.12	86.21±0.60	3.63±0.11	64.99±0.76	4.66±0.32	40.08±0.85
ER	77.69±1.45	25.96±1.68	54.40±1.12	35.65±1.96	17.13±1.21	50.22±1.20	12.36±1.22	38.59±0.83
RE-ER	78.58±1.02	24.84±1.28	54.99±2.72	34.11±4.52	15.95±0.96	51.39±1.45	12.42±1.18	37.68±2.03
MER	82.46±0.83	17.36±0.83	–	–	–	–	–	–
AGEM	40.03±7.02	73.86±8.73	17.62±0.47	81.02±3.32	3.12±0.25	59.90±0.86	4.30±0.30	**35.62**±1.25
GSS	62.06±1.93	45.50±2.33	39.62±2.14	52.63±2.53	6.67±0.09	53.79±0.75	–	–
DER	84.36±0.79	15.25±1.07	36.17±2.62	54.96±5.28	3.74±0.20	65.15±1.09	5.12±0.29	41.92±0.85
DER++	86.43±0.88	15.97±1.15	55.32 ±1.08	26.37±3.08	16.09±0.89	50.83±0.79	12.83±0.58	38.34±0.06
CLS-ER	84.73±1.02	17.04±1.30	54.12±2.59	34.89±2.73	16.72±1.72	49.39±2.17	11.31±0.38	40.28±0.99
CN	–	–	54.08 ±2.24	27.09±5.01	16.01±0.55	48.95 ±0.32	11.51±0.44	42.28±0.63
TRAF	**86.95**±0.81	**14.57**±1.05	**59.58**±1.02	**23.95**±2.66	**20.53**±1.74	**48.05**±1.83	**14.37**±1.05	38.11±1.28

The bold values denote the best performance
'–' indicates experiments we were unable to run, due to compatibility issues or intractable training time. [↑] Higher is better and [↓] lower is better

our method. However, the ACC of AGEM is significantly lower than that of ours (10% lower).

Comparisons with baselines on the setting of Task-Aware Table 4 show the ACC and FT results in the protocol of *Task-Aware*. Similar to the setting of *Task-Free*, the proposed method could achieve higher performance with considerable forgetting than other methods. For instance, on the Split CIFAR100, the ACC of TRAF is 75.00%, largely higher than the best performance of baselines, i.e., 70.47%. On the Split TinyImageNet, the forgetting (FT) of TRAF is better than other methods, except for UCL. However, for UCL, its ACC is significantly lower than ours (over 30%). It is because that UCL uses the unsupervised learning loss to train the network and requires sufficient learning to learn the representations well. Thus, it performs poorly in the online fashion.

The accuracy curves Fig. 4 shows the curves of average accuracy evaluated on the observed tasks when the network has been trained on each task on the datasets of Split CIFAR100 and Split TinyImageNet, respectively. According to Fig. 4, the performance of our method is higher than baselines continually, further validating the superiority of the proposed method in alleviating the stability-plasicity dilemma.

Combining with more CL methods As shown in Tables 5, when combined with our method, the performance of combining methods sometimes could be higher than TRAF. For

Table 4 Results of ACC (%) [↑] and forgetting (%) [↓] evaluated on all tasks after finishing learning all tasks on the setting of *Task-Aware*

Method	S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]
SGD	95.73±1.32	4.14±1.66	72.25±1.42	19.27±2.27	34.03±3.22	33.02±3.75	25.59±1.31	16.91±1.61
ER	98.44±0.15	0.67±0.20	86.77±0.75	2.87±0.41	69.91±1.39	5.09±0.97	47.88±1.60	5.28±1.02
RE-ER	98.31±0.39	0.81±0.46	87.22±0.68	2.08±0.47	69.61±0.76	5.07±1.16	47.33±2.05	5.73±1.93
MER	97.47±0.35	0.88±0.35	–	–	–	–	–	–
AGEM	98.49±0.33	0.84±0.43	70.78±3.53	15.31±1.87	41.50±2.39	20.10±2.57	32.08±1.66	5.86±1.67
GSS	97.85±0.15	1.30±0.17	80.41±2.08	9.84±2.79	56.65±2.02	14.52±1.64	–	–
DER	98.70±0.11	0.68±0.15	82.53±0.79	3.99±1.01	42.82±2.58	24.08±1.73	31.73±1.75	12.58±1.99
DER++	98.81±0.07	0.61±0.11	87.48±1.05	1.74±0.73	70.47±1.38	4.11±0.60	49.15±0.18	4.77±0.47
UCL	–	-	67.72±1.03	1.67±1.19	43.54±1.36	3.47±1.36	20.78±0.57	**1.32**±0.20
CLS-ER	98.62±0.15	0.58±0.19	87.04±1.81	1.95±2.05	70.23±0.77	3.53±0.53	46.64±0.94	6.44 ±0.49
CN	–	–	86.55±1.26	2.25±1.10	68.46±1.39	4.58±0.76	47.31±0.44	7.24±0.38
TRAF	98.92±0.13	**0.46**±0.17	**89.65**±0.35	**1.04**±0.42	**75.00**±0.88	**3.32**±0.35	**51.91**±1.72	4.16±1.46

The bold values denote the best performance
'–' indicates experiments we were unable to run, due to compatibility issues or intractable training time. [↑] Higher is better and [↓] lower is better

Table 5 Results of ACC (%) [↑] evaluated on all tasks after finishing learning all tasks on the setting of *Task-Aware* (*T-Aware*) and *Task-Free* (*T-Free*)

Method	S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
	T-Free	T-Aware	T-Free	T-Aware	T-Free	T-Aware	T-Free	T-Aware
oEWC (Schwarz et al., 2018)	20.37±0.23	97.97±0.67	18.70±0.07	66.91±5.39	3.35±0.08	30.88±1.46	2.80±0.31	10.93±0.63
oEWC + TRAF	86.09±1.45	98.38±0.71	37.66±1.68	80.32±0.36	17.59±1.90	69.02±1.80	11.73±1.36	47.07±1.36
SI (Zenke et al., 2017)	19.91±0.17	97.00±0.95	18.21±0.25	61.52±3.03	3.40±0.07	29.78±1.93	3.89±0.10	17.40±1.73
SI +TRAF	**87.56**±1.04	98.51±0.15	53.23±1.93	88.15±0.85	19.10±1.56	74.50±1.27	13.78±0.84	**52.97**±1.05
TRAF (Ours)	86.95±0.81	**98.92**±0.13	**59.58**±1.02	**89.65**±0.35	**20.53**±1.74	**75.00**±0.88	**14.37**±1.05	51.91±1.72

The bold values denote the best performance

Table 6 Results of ACC (%) [↑] evaluated on all tasks after finishing learning all tasks on the setting of *Task-Free* in online continual learning. 2 Split, 5 Split, 10 Split, and 20 Split divide all 100 classes of CIRAR-100 into 2, 5, 10, and 20 splits, respectively

Method	2 Splits	5 Splits	10 Splits	20 Splits
DER'	10.99±0.25	13.91±0.12	16.23±0.09	17.09±0.25
FOSTER	12.54±0.51	13.05±0.37	10.24±1.66	7.60±0.71
TRAF (Ours)	**26.02**±0.81	**22.66**±1.49	**22.93**±0.57	**20.53**±0.81

The bold values denote the best performance
The classes of each task are disjoint. The memory size is 1000. [↑] Higher is better

Table 7 Results of ACC (%) and FT (%)

Method	5-Split CIFAR-10		5-Split CIFAR-100	
	ACC [↑]	FT [↓]	ACC [↑]	FT [↓]
RM	65.88±1.80	31.97±4.37	21.55±0.60	17.40±0.84
CCS	**66.06**±0.99	**30.76**±5.75	**22.36**±0.43	**10.38**±0.15

The bold values denote the best performance
The experiments run three times. The memory size for Split CIFAR-10 and Split CIFAR-100 are 500 and 1000, respectively. 5-Split CIFAR-10 and 5-Split CIFAR-100 divide the total classes of CIFAR-10 and CIFAR-100 into five tasks, respectively. The classes of each task are disjoint

example, SI+TRAF could obtain higher performance on S-MNIST (*Task Free*) and S-TinyImageNet (*Task Aware*).

Comparison with more recent works According to Table 6, compared to DER' (Yan et al., 2021) and FOSTER (Wang et al., 2022a), our proposed method achieves the best performance with a significant margin. Moreover, according to Table 7, CCS can achieve better performance and lower forgetting than rainbow memory (RM), validating the superiority of CCS in selecting data.

Table 8 Effect of each component

Module				S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
S-P	I-P/D	CCS	DKD	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]
✓				79.70	23.64	47.92	41.59	12.98	57.95	12.28	39.27
✓	✓			81.67	21.36	56.08	29.36	19.77	48.86	13.08	39.06
✓	✓	✓		82.96	19.58	57.26	27.15	20.08	48.34	13.96	38.81
✓	✓	✓	✓	**86.95**	**14.57**	**59.58**	**23.95**	**20.53**	**48.05**	**14.37**	**38.11**

The bold values denote the best performance
The experiments are average results of five runs on the setting of *Task-Free*. "I-P/D" denotes intra-process without DKD

Table 9 Ablation studies for t-test

	S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]
t-test	79.07	24.66	43.78	46.55	16.38	53.54	12.66	40.23
Ours	**81.67**	**21.36**	**56.08**	**29.36**	**19.77**	**48.86**	**13.08**	**39.06**

The bold values denote the best performance
The experiments run five times

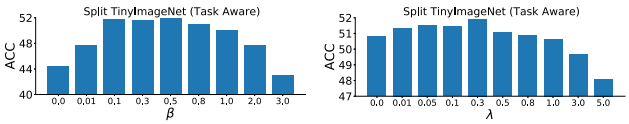


Fig. 5 The effect of β and λ

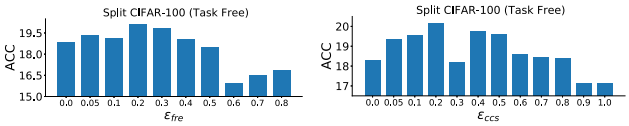


Fig. 6 The effect of ϵ_{fre} and ϵ_{ccs} in trust-region adaptive frequency

5.3 Ablation Study and Analysis

Effect of each component We show the effect of each component. According to Table 8, both adding I-P and CCS could obtain higher ACCs and lower FTs. Especially, the ACC of adding I-P is 8.16% higher and the FT is 12.23% lower than S-P on the S-CIFAR-10, indicating the effectiveness of trust-region adaptive frequency. Similarly, adding the component of confidence-based coreset selection can achieve better stability-plasticity trade-off and less forgetting on all benchmarks. For example, on S-CIFAR-10, the improvement of adding CCS is 1.16% and 2.21% for accuracy and forgetting, respectively. Moreover, combing all components can further improve the performance and decrease forgetting.

Effect of β and λ Fig. 5 shows that both too large a value or too small a value of the balance factor β result in poor performance. If β is too large, the model will pay too much attention to preventing forgetting, resulting in unsatisfactory learning of the current task. However, if β is too small, the model could not retain the past knowledge well, resulting in catastrophic forgetting. Similarly, too large or too small a value of the distillation factor λ will also result in an improper balance between the learning of coreset and knowledge distillation, leading to worse stability-plasticity trade-off.

The effect of ϵ_{fre} and ϵ_{ccs} As shown in Fig. 6, when ϵ_{fre} is close to zero, the interval is almost $inv_{\min}$, then the intra-process will be performed frequently at the beginning of learning of each task, where the data in coreset are most from old tasks, leading to worse performance. Or if ϵ_{fre} is large, the interval is almost $inv_{\max}$ and the frequency is low. Then the model could not exploit the advantage of intra-process. Therefore, ϵ_{fre} needs to be proper to achieve better performance. We also explore the effect of ϵ_{ccs} in Fig. 6. According to Fig. 6, when $\epsilon_{\text{ccs}} = 1.0$, the model randomly selects data from the current batch. The selection strategy degenerates into *reservoir*, and the performance is worse. However, when $\epsilon_{\text{ccs}} = 0.0$, the model selects the examples with higher confidence scores all the time, the performance is also unsatisfactory because the confidence scores are not reliable when the model does not learn well.

Comparison with t-test Table 9 show the results of using t-test rule in Eq. (3). The experiments run five times. According to Table 9, we could find that using the average score is better than the t-test, validating the superiority of using the proposed rule.

The result of combining into one loss Table 10 shows the results of the baseline (OneLoss) that combines the two processes into one loss and optimizes it at every iteration. The results show that our method performs better than OneLoss, validating the essential of alternating between standard-process and intra-process.

Ablation studies on rules The results in Table 11 show that using the average score is better than all other strategies, validating the reasonability of using the average operation.

Table 10 Results of ACC (%) evaluated on all tasks after finishing learning all tasks. Higher is better

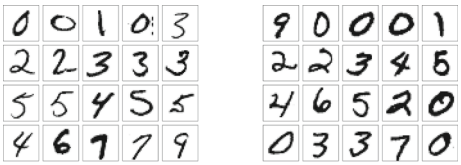
Method	S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
	T-Free	T-Aware	T-Free	T-Aware	T-Free	T-Aware	T-Free	T-Aware
OneLoss	77.83±1.79	98.50±0.17	36.53±2.43	86.20±2.42	13.59±1.46	71.79±0.55	13.00±0.23	49.31±1.36
Ours	**86.95**±0.81	**98.92**±0.13	**59.58**±1.02	**89.65**±0.35	**20.53**±1.74	**75.00**±0.88	**14.37**±1.05	**51.91**±1.72

The bold values denote the best performance

Table 11 Ablation studies on rules

		S-MNIST		S-CIFAR-10		S-CIFAR-100		S-TinyImageNet	
		ACC[↑]	FT[↓]	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]	ACC[↑]	FT[↓]
ϵ_{fre}	Avg/std	80.70	22.57	52.51	36.03	18.70	49.91	12.66	39.29
	Std. dev	80.45	22.91	55.23	30.41	18.18	50.32	12.53	40.54
	Random	80.68	22.52	49.43	37.86	17.93	51.15	12.53	40.54
	Ours	**81.67**	**21.36**	**56.08**	**29.36**	**19.77**	**48.86**	**13.08**	**39.06**
ϵ_{mem}	Avg/std	81.09	22.12	50.08	37.09	19.00	48.47	11.05	42.79
	Std. dev	79.73	23.81	52.65	30.56	18.79	49.11	11.38	42.10
	Random	79.57	23.91	53.60	31.32	19.15	48.48	13.78	39.76
	Ours	**82.96**	**19.58**	**57.26**	**27.15**	**20.08**	**48.34**	**13.96**	**38.81**

The bold values denote the best performance
[↑] Higher is better and [↓] lower is better



(a) Random (b) CCS

Fig. 7 The data comparison of selecting by random strategy and Confidence-based Coreset Selection (CCS)

5.4 Selected Data

Figure 7 shows the comparison of randomly selected data and the data chosen by the proposed confidence-based coreset selection. We could find that the data selected by CSS are bolder and easier to distinguish the true class, i.e., more representative, validating the effectiveness of our selection strategy.

5.5 Running Time

Figure 8 shows the comparison of running time. The device is a single Nvidia Tesla V100 (16GB) GPU. The dataset is the Split CIFAR-100, and the results are the average results over five runs. According to Fig. 8, since our proposed method alternates the intra-process and standard-process, the running time is marginally larger than some baselines. However, we can find that the running time of the proposed method is also lower than some methods, e.g., GSS.

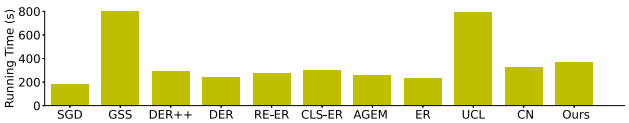


Fig. 8 Comparison of running time (s). The device is a single Nvidia Tesla V100 (16GB) GPU. The dataset is the Split CIFAR-100 and the results are the average results over five runs. For a better comparison, we only show the range between [0, 800]

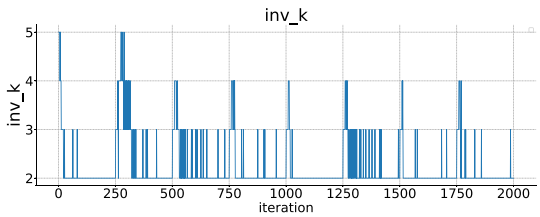


Fig. 9 The changes of inv_k during training. The dataset is Split CIFAR-100. The smaller inv_k, the higher the frequency of intra-process is

5.6 The Changes of inv_k During Training

Figure 9 shows how the frequency inv_k involves overtime. The larger inv_k, the higher the frequency of intra-process is. Since the batch size is 10 and the datasize of one task is 2500. Therefore, new tasks arrive every 250 iterations. According to Fig. 9, we could find that inv_k would increase the frequency at the beginning of each task. And at the later stage of training of each task, the coreset store more data of the current task and as shown in Fig. 3, the coreset becomes more balanced. Therefore, inv_k decreases, i.e., the frequency of intra-process increases. The result shows that the frequency of the intra-

process can be adjusted dynamically based on the learning stage, validating the effect of our method.

6 Conclusion

In this paper, we aim to relieve the stability-plasticity dilemma for continual learning, constraining that the data arrives in an online stream. We propose a new online continual learning approach, Trust-Region Adaptive Frequency (TRAF), which alternates between standard-process and intra-process updates in an adaptive frequency. Moreover, TRAF also retains useful knowledge through dark knowledge distillation and stores representative data based on confidence scores. Extensive experimental results validate the effectiveness of the proposed method on several benchmarks. For limitation, the proposed method is tailed for the online setting and may not show excellence in other continual learning settings, e.g., the offline setting. We would like to explore more realistic and challenging scenarios and more analytical analysis in the future. Moreover, studying other continual learning methods, e.g., regularization-based methods, in the online setting is also an interesting direction.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability The datasets generated during and/or analysed during the current study are available in the MNIST http://yann.lecun.com/exdb/mnist/, CIFAR-10, CIFAR-100 https://www.cs.toronto.edu/~kriz/cifar.html, and TinyImageNet https://drive.google.com/u/0/uc?id=1Sy3ScMBr0F4se8VZ6TAwDYF-nNGAAdxj&export=download repositories.

References

Aljundi, R., Chakravarty, P., & Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3366–3375).

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., & Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 139–154).

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., & Page-Caccia, L. (2019a). Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems* (pp. 11849–11860).

Aljundi, R., Lin, M., Goujaud, B., & Bengio, Y. (2019b). Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, (pp. 11816–11825).

Arani, E., Sarfraz, F., & Zonooz, B. (2022). Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations (ICLR)*.

Bang, J., Kim, H., Yoo, Y., Ha, J. W., & Choi, J. (2021). Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 8218–8227).

Buzzega, P., Boschini, M., Porrello, A., Abati, D., & Calderara, S. (2020). Dark experience for general continual learning: A strong, simple baseline. arXiv:2004.07211.

Buzzega, P., Boschini, M., Porrello, A., & Calderara, S. (2021). Rethinking experience replay: A bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE (pp. 2180–2187).

Cartis, C., Gould, N. I., & Toint, P. L. (2011). Adaptive cubic regularisation methods for unconstrained optimization. Part I: Motivation, convergence and numerical results. *Mathematical Programming, 127*(2), 245–295.

Chaudhry, A., Dokania, P. K., Ajanthan, T., & Torr, P. H. (2018a). Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 532–547).

Chaudhry, A., Ranzato, M., Rohrbach, M., & Elhoseiny, M. (2018b). Efficient lifelong learning with a-gem. arXiv:1812.00420.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., & Ranzato, M. (2019). Continual learning with tiny episodic memories. In *International conference on machine learning Workshop: Multi-Task and Lifelong Reinforcement Learning*.

Conn, A. R., Gould, N. I., & Toint, P. L. (2000). Trust region methods. *Society for Industrial and Applied Mathematics (SIAM), Philadelphia. vol 1*

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., & Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*, 3366–3385.

Ghadimi, S., & Lan, G. (2013). Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization, 23*(4), 2341–2368.

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision, 129*(6), 1789–1819.

Grossberg, S. (1982). How does a brain build a cognitive code? In *Studies of Mind and Brain: Neural principles of learning, perception, development, cognition, and motor control* (pp. 1–52). Springer.

He, K., Zhang ,X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Hou, S., Pan, X., Loy, C. C., Wang, Z., & Lin, D. (2018). Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 437–452).

Hou, S., Pan, X., Loy, C. C., Wang, Z., &Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp 831–839).

Isele, D., & Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 32*, 3302–3309.

Jerfel, G., Grant, E., Griffiths, T., &Heller, KA. (2019). Reconciling meta-learning and continual learning with online mixtures of tasks. In *Advances in Neural Information Processing Systems* (pp. 9122–9133).

Jin, X., Sadhu, A., Du, J., & Ren, X. (2021). Gradient-based editing of memory examples for online task-free continual learning. 2006.15294.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521–3526.

Kong, Y., Liu, L., Wang, Z., & Tao, D. (2022). Balancing stability and plasticity through advanced null space in continual learning. In*European Conference on Computer Vision*.

Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*. Citeseer: Princeton.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Lee, S. W., Kim, J. H., Jun, J., Ha, J. W., & Zhang, B. T. (2017). Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems* (pp. 4652–4662).

Li, X., Zhou, Y., Wu, T., Socher, R., & Xiong, C. (2019). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. arXiv:1904.00310.

Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*(12), 2935–2947.

Lin, S., Yang, L., Fan, D., & Zhang, J. (2022). Trgp: Trust region gradient projection for continual learning. 2202.02931.

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

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems* (pp. 6467–6476).

Madaan, D., Yoon, J., Li, Y., Liu, Y., & Hwang, S. J. (2022). Representational continuity for unsupervised continual learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=9Hrka5PA7LW.

Mallya, A., & Lazebnik, S. (2018). 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).

Mallya, A., Davis, D., & Lazebnik, S. (2018). 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).

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, *vol 24* (pp 109–165). Elsevier.

Mermillod, M., Bugaiska, A., & Bonin, P. (2013). The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, vol. 4, 2013, Art. no. 504.

Mirzadeh, S. I., Farajtabar, M., Pascanu, R., & Ghasemzadeh, H. (2020). Understanding the role of training regimes in continual learning. In *Advances in Neural Information Processing Systems (NeurIPS), 33*, 7308–7320.

Nguyen, C. V., Li, Y., Bui, T. D., & Turner, R. E. (2017). Variational continual learning. arXiv:1710.10628.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Berlin: Springer.

Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., & Nabi, M. (2019). Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11321–11329).

Park, D., Hong, S., Han, B., & Lee, K. M. (2019). Continual learning by asymmetric loss approximation with single-side overestimation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 3335–33440).

Peng, Y., Qi, J., Ye, Z., & Zhuo, Y. (2021). Hierarchical visual-textual knowledge distillation for life-long correlation learning. *International Journal of Computer Vision, 129*(4), 921–941.

Pham, Q., Liu, C., & Hoi, S. (2021). Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, *34*.

Pham, Q., Liu, C., & Steven, H. (2022). Continual normalization: Rethinking batch normalization for online continual learning. In *ICLR*.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., & Hadsell, R. (2019). Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems* (pp. 7647–7657).

Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review, 97*(2), 285.

Rebuffi, S. A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2001–2010).

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., & Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. arXiv:1810.11910.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., & Tesauro, G. (2019). Learning to learn without forgetting by maximizing transfer and minimizing interference. 1810.11910.

Ritter, H., Botev, A., & Barber, D. (2018). Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems* (pp. 3738–3748).

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., & Wayne, G. (2019). Experience replay for continual learning. In *Advances in Neural Information Processing Systems* (pp. 350–360).

Rusu, AA., Rabinowitz, NC., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. arXiv:1606.04671.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., & Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, PMLR (pp. 4528–4537).

Serra, J., Suris, D., Miron, M., & Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, PMLR, (pp. 4548–4557).

Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems* (pp. 2990–2999).

Stanford. (2015). Tiny ImageNet Challenge (CS231n). http://tiny-imagenet.herokuapp.com/.

Sun, S., Calandriello, D., Hu, H., Li, A., & Titsias, M. (2022). Information-theoretic online memory selection for continual learning. In *International Conference on Learning Representations*.

Tang, S., Chen, D., Zhu, J., Yu, S., & Ouyang, W. (2021). Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9634–9643).

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS), 11*(1), 37–57.

Wang, F. Y., Zhou, D. W., Ye, H. J., & Zhan, D. C. (2022a). Foster: Feature boosting and compression for class-incremental learning. arXiv:2204.04662.

Wang, Z., Liu, L., & Tao, D. (2020). Deep streaming label learning. *International Conference on Machine Learning (ICML), 119*, 9963–9972.

Wang, Z., Liu, L., Duan, Y., Kong, Y., & Tao, D. (2022b). Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 171–181).

Wang, Z., Liu, L., Duan, Y., & Tao, D. (2022). Continual learning through retrieval and imagination. *Proceedings of the AAAI Conference on Artificial Intelligence, 36*(8), 8594–8602. https://doi.org/10.1609/aaai.v36i8.20837. ojs.aaai.org/index.php/AAAI/article/view/20837.

Wang, Z., Liu, L., Kong, Y., Duan, Y., & Tao, D. (2022d). Online continual learning with contrastive vision transformer. In *European Conference on Computer Vision*.

Wu, L., Liu, B., Stone, P., & Liu, Q. (2020). Firefly neural architecture descent: A general approach for growing neural networks. In *Advances in Neural Information Processing Systems (NeurIPS), 33*, 22373–22383.

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

Yin, D., Farajtabar, M., Li, A., Levine, N., & Mott, A. (2020). Optimization and generalization of regularization-based continual learning: A loss approximation viewpoint. 2006.10974.

Yoon, J., Kim, S., Yang, E., & Hwang, S. J. (2019). Scalable and order-robust continual learning with additive parameter decomposition. arXiv:1902.09432.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, PMLR, (pp. 3987–3995).

Zhao, T., Han, J., Yang, L., Wang, B., & Zhang, D. (2021). Soda: Weakly supervised temporal action localization based on astute background response and self-distillation learning. *International Journal of Computer Vision, 129*(8), 2474–2498.

Zhou, G., Sohn, K., & Lee, H. (2012). Online incremental feature learning with denoising autoencoders. In *Artificial Intelligence and Statistics* (pp. 1453–1461).