# Online Class-Incremental Continual Learning with Adversarial Shapley Value

**Dongsub Shim**[1*], **Zheda Mai**[1*], **Jihwan Jeong**[1*], **Scott Sanner**[1]
**Hyunwoo Kim**[2], **Jongseong Jang**[2]
[1] University of Toronto, [2] LG AI Research
(dongsub.shim, zheda.mai)@mail.utoronto.ca, (jhjeong, ssanner)@mie.utoronto.ca, (hwkim, j.jang)@lgresearch.ai

**Abstract**

As image-based deep learning becomes pervasive on every device from cell phones to smart watches, there is a growing need to develop methods that continually learn from data while minimizing memory footprint and power consumption. While memory replay techniques have shown exceptional promise for this task of *continual learning*, the best method for selecting which buffered images to replay is still an open question. In this paper, we specifically focus on the online class-incremental setting where a model needs to learn new classes continually from an online data stream. To this end, we contribute a novel Adversarial Shapley value scoring method that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes (to maintain learning stability and avoid forgetting) while interfering with latent decision boundaries of current classes being learned (to encourage plasticity and optimal learning of new class boundaries). Overall, we observe that our proposed ASER method provides competitive or improved performance compared to state-of-the-art replay-based continual learning methods on a variety of datasets.

## 1 Introduction

Image-based deep learning is a pervasive but computationally expensive and memory intensive task. Yet the need for such deep learning on personal devices to preserve privacy, minimize communication bandwidth, and maintain real-time performance necessitates the development of methods that can continuously learn from streaming data while minimizing memory storage and computation footprint. However, a well-documented defect of deep neural networks that prevents it from learning continually is called *catastrophic forgetting* (McCloskey and Cohen 1989) — the inability of a network to perform well in previously seen tasks after learning new tasks. To address this challenge, the field of *continual learning* (CL) studies the problem of learning from a (non-iid) stream of data, with the goal of preserving and extending the acquired knowledge over time.

Many existing CL approaches use a *task incremental* setting where data arrives one task (i.e., set of classes to be identified) at a time and the model can utilize task identity during both training and testing (Kirkpatrick et al. 2017; Li and Hoiem 2016; Lopez-Paz and Ranzato 2017). Specifically, a common practice in this setting is to assign a separate output layer (head) for each task; then the model just needs to classify labels within a task, which is known as multi-head evaluation (Chaudhry et al. 2018). However, this setting requires additional supervisory signals at test time — namely the task identity — to select the corresponding head, which obviates its use when the task label is unavailable. In this work, we consider a more realistic but difficult setting, known as *online class-incremental*, where a model needs to learn new classes continually from an online data stream (each sample is seen only once). In contrast to the task incremental setting, this setting adopts the single-head evaluation, where the model needs to classify all labels without task identity. Moreover, we focus on image classification, a common application where this setting is used (Aljundi et al. 2019a,b; Lee et al. 2020; Chrysakis and Moens 2020).

Current CL methods can be taxonomized into three major categories: regularization-based, parameter isolation, and memory-based methods (Parisi et al. 2019; De Lange et al. 2019). Regularization-based methods incorporate an additional penalty term into the loss function to penalize the update of critical model parameters (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017; Aljundi et al. 2018; Ritter, Botev, and Barber 2018). Other regularization-based methods imposed knowledge distillation techniques to penalize the feature drift on previous tasks (Li and Hoiem 2016; Wu et al. 2019; Rannen et al. 2017). Parameter isolation methods assign per-task parameters to bypass interference by expanding the network and masking parameters to prevent forgetting (Mallya and Lazebnik 2018; Lee et al. 2020; Yoon et al. 2018). Memory-based methods deploy a memory buffer to store a subset of data from previous tasks. The samples from the buffer can be either used to constrain the parameter updates such that the loss on previous tasks cannot increase (Chaudhry et al. 2019a; Lopez-Paz and Ranzato 2017), or simply for replay to prevent forgetting (Rebuffi et al. 2017; Chaudhry et al. 2019b).

Regularization methods only protect the model's ability to classify within a task and thus they do not work well in our setting where the ability to discriminate among classes from different tasks is crucial (Lesort, Stoian, and Filliat 2019). Also, most parameter isolation methods require task identity

*Authors contributed equally.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

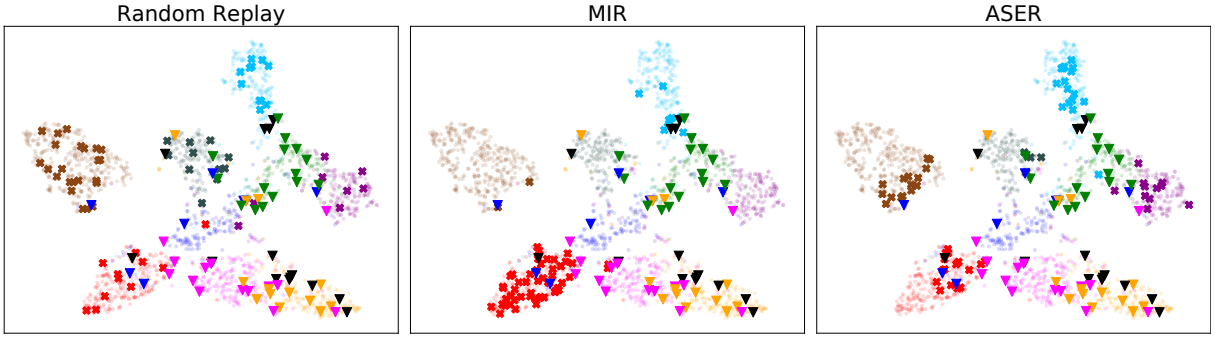



Figure 1: 2D t-SNE (van der Maaten and Hinton 2008) visualization of CIFAR-100 data embeddings and their class labels (different colors) showing current task samples (triangle), memory samples (pale circle) and retrieved memory samples for rehearsal (bold x). For each point, we obtain the latent embedding from reduced ResNet18 (Chaudhry et al. 2019a). Note that Random Replay (Chaudhry et al. 2019b) distributes its retrieved samples non-strategically, MIR disproportionately selects seemingly redundant samples in a single – apparently most interfered – class (red), whereas ASER strategically retrieves memory samples that are representative of different classes in memory but also adversarially located near class boundaries and current task samples.

during inference, which violates our setting. Therefore in this work, we consider the replay approach which has shown to be successful and efficient for the online class-incremental setting (Aljundi et al. 2019b,a). Since the memory buffer is the only place to store data from previous tasks, a key question for replay-based methods is *how to update and retrieve memory samples when new data arrives?* For example, Chaudhry et al. (2019b) proposed a simple but strong baseline that randomly updates and retrieves samples, while the highly effective Maximally Interfered Retrieval (MIR) method (Aljundi et al. 2019a) chooses replay samples whose loss most increases after a current task update. However, if we visualize the latent space of retrieved memory samples chosen by each method in Figure 1, we see that the methods mentioned above fail to strategically select samples that both preserve existing memory-based class boundaries while protecting against current task samples that interfere with these boundaries (detailed discussion in caption of Figure 1).

We address the deficiencies observed above by proposing a novel replay-based method called Adversarial Shapley value Experience Replay (ASER). ASER is inspired by the Shapley value (SV) (Shapley 1953) used in cooperative game theory to fairly distribute total gains to all players — in our CL setting, we use the SV to determine the contribution of memory samples to learning performance (Ghorbani and Zou 2019; Jia et al. 2019a,b). We also introduce an adversarial perspective of SV for CL memory retrieval that aims to score memory samples according to their preservation of decision boundaries for "friendly" samples in the memory buffer (to maintain learning stability and avoid forgetting) and their interference with "opponent" samples from the current task that disrupt existing memory-based class boundaries (to encourage plasticity and optimal learning). Through extensive experiments on three commonly used benchmarks in the CL literature, we demonstrate that ASER provides competitive or improved performance compared to state-of-the-art replay-based methods, especially when the memory buffer size is small.

## 2 Continual Learning

### Problem Definition

**Online Class-Incremental Learning** Following the recent CL literature (Aljundi et al. 2019a,b; Lee et al. 2020; Chrysakis and Moens 2020), we consider the online supervised class-incremental learning setting where a model needs to learn new classes continually from an online data stream (each sample is seen only once). More concretely, a neural network classifier $f : \mathbb{R}^d \mapsto \mathbb{R}^C$, parameterized by $\theta$ will receive input batches $B_n^t$ of size $b$ from task$_t$. Task$_t$ consists of classes that the classifier has never seen before in task$_{1:t}$. Moreover, we adopt the single-head evaluation setup (Chaudhry et al. 2018) where the classifier has no access to task identity during inference and hence must choose among all labels. Our goal is to train the classifier $f$ to continually learn new classes from the data stream without forgetting.

**Metrics** Since the goal of CL is to continually acquire new knowledge while preserving existing learning, we use two standard metrics in the CL literature to measure performance: *average accuracy* for overall performance and *forgetting* to measure how much acquired knowledge the algorithm has forgotten (Chaudhry et al. 2018, 2019b). In **Average Accuracy**, $a_{i,j}$ is the accuracy evaluated on the held-out test set of task $j$ after training the network from task 1 to $i$. In **Average Forgetting**, $f_{i,j}$ represents how much the model forgets about task $j$ after being trained on task $i$. For $T$ tasks:

$$\text{Average Accuracy}(A_T) = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}$$

$$\text{Average Forgetting}(F_T) = \frac{1}{T-1}\sum_{j=1}^{T-1} f_{T,j}$$

$$\text{where} \quad f_{i,j} = \max_{l\in\{1,\cdots,i-1\}} a_{l,j} - a_{i,j}$$

### Experience Replay Methods

**Experience Replay (ER)** The research of ER and the important role of replay buffers has been well-established in the reinforcement learning area (Rolnick et al. 2019; Foerster et al. 2017). Recently, ER has been widely applied in supervised CL learning tasks (Reimer et al. 2019; Aljundi et al. 2019a,b). Compared with the simplest baseline model that fine-tunes the parameters based on the new task without any measures to prevent forgetting, ER makes two simple modifications: (1) it stores a subset of the samples from past tasks in a memory buffer $\mathcal{M}$ of limited size $M$; (2) it concatenates the incoming minibatch $B_n$ with another minibatch $B_{\mathcal{M}}$ of samples selected from the memory buffer. Then, it simply takes a SGD step with the combined batch, followed by an online update of the memory. A generic ER algorithm is presented in Algorithm 1.

What differentiates various replay-based methods are the *MemoryRetrieval* in line 3 and the *MemoryUpdate* in line 5. Although there exists another stream of replay methods that utilize a generative model to produce virtual samples instead of using a memory buffer (Shin et al. 2017), recent research has demonstrated the limitations of such approaches with convolutional neural networks in datasets such as CIFAR-10 (Aljundi et al. 2019a; Lesort et al. 2018). Hence, we focus on the memory-based approach in our work.

Basic **ER** is a simple but strong baseline that applies reservoir sampling in *MemoryUpdate* and random sampling in *MemoryRetrieval*. Despite its simplicity, recent research has shown that naive ER outperforms many specifically designed CL approaches with and without a memory buffer (Chaudhry et al. 2019b).

**Algorithm 1:** Generic ER-based method

**Input :** Batch size $b$, Learning rate $\alpha$
**Initialize:** Memory $\mathcal{M} \leftarrow \{\} * M$; Parameters $\theta$; Counter $n \leftarrow 0$

1 **for** $t \in \{1, \ldots, T\}$ **do**
2  **for** $B_n \sim D_t$ **do**
3  $B_{\mathcal{M}} \leftarrow$ *MemoryRetrieval*$(B_n, \mathcal{M})$
4  $\theta \leftarrow \text{SGD}(B_n \cup B_{\mathcal{M}}, \theta, \alpha)$
5  $\mathcal{M} \leftarrow$ *MemoryUpdate*$(B_n, \mathcal{M})$
6  $n \leftarrow n + b$
7 **return** $\theta$

**Maximally-interfered Retrieval (MIR)** MIR is a recently proposed method aiming to improve the *MemoryRetrieval* strategy (Aljundi et al. 2019a). MIR chooses replay samples according to loss increases given the estimated parameters update based on the newly arrived data. However, samples with significant loss increases tend to be similar in the latent space, which may lead to redundancy in the retrieved data, as shown in Figure 1. Like ER, MIR uses reservoir sampling for the *MemoryUpdate*.

**Gradient-based Sample Selection (GSS)** Different from MIR, GSS pays attention to the *MemoryUpdate* strategy (Aljundi et al. 2019b). Specifically, it tries to diversify the gradients of the samples in the memory buffer. Like ER, GSS uses random sampling in *MemoryRetrieval*.

## 3 Efficient Computation of Shapley Value via KNN Classifier

When we return to Figure 1 and analyze the latent embeddings of memory samples, we observe the natural clustering effect of classes in the embedding space, which has been well-observed previously in the deep learning literature (Min et al. 2009; Donahue et al. 2014). On account of this, we observe that some samples may indeed be more important than others in terms of preserving what the neural network has learned. For example, data from one class that are near the boundary with data from another class in some sense act as sentinels to guard the decision boundary between classes. This suggests the following question: *how can we value data in the embedded space in terms of their contribution to accurate classification?*

Given that the embedding plot of Figure 1 suggests that a new data point is likely to take the classification of its nearest neighbors in the embedding space, we could rephrase this question as asking how much each data point in memory contributes to correct classification from the perspective of a K-Nearest Neighbors (KNN) classifier. Fortunately, the existing research literature already provides both a precise and efficient answer to this question viewed through the lens of Shapley data valuation for KNN classifiers (Jia et al. 2019a; Ghorbani and Zou 2019; Jia et al. 2019b). Before we cover this solution, we first pause to recap the purpose of Shapley values.

**Shapley Value (SV) for Machine Learning** The SV (Shapley 1953; Roth 1988) was originally proposed in cooperative game theory to decide the share of total gains for each player in a coalition. The SV has a set of mathematical properties that make it appealing to many applications: *group rationality*, *fairness*, and *additivity*. Conversely, it can be shown that the SV is the *only* allocation scheme that satisfies these three properties.

In the context of machine learning, the SV has been used to estimate the individual contribution of data points to the performance of a trained model *in the context of all other data* (Ghorbani and Zou 2019; Jia et al. 2019b). Formally, let $N$ denote the number of data points and $I = \{1, \ldots, N\}$ be the associated index set. Then, each datum is interpreted as a player of a cooperative game with the goal of maximizing test-time performance. Let $v(S)$ define a utility function of the ML model over a subset $S \subset I$ on which the model is trained. Then, the SV of a data point of index $i$ with the utility $v(S)$ is the following:

$$s(i) = \frac{1}{N} \sum_{S \subseteq I \setminus \{i\}} \frac{1}{\binom{N-1}{|S|}} [v(S \cup \{i\}) - v(S)] \qquad (1)$$

Intuitively, when we consider every possible subset of data points, $s(i)$ measures the average marginal improvement of

utility given by the sample $i$. By setting the utility as test accuracy in ML classification tasks, the SV can discover how much of the test accuracy is attributed to a training instance.

**Efficient KNN Shapley Value Computation** Specific to our requirements for data valuation in this paper, recent work has developed an efficient method for SV computation in a KNN classification framework (Jia et al. 2019a). This is a critical innovation since the direct powerset-based computation of the SV requires $O(2^N)$ evaluations for general, bounded utility functions. Furthermore, each evaluation involves training an ML model with a given subset of data ($S$). This is prohibitive in most modern deep learning applications, not to mention online CL with neural networks. As shown in (Jia et al. 2019a) and summarized below, the exact KNN-SV can be computed in $O(N \log N)$.

Let $(\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}})$ denote an *evaluation point* and $D_c = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_c}$ a *candidate set*, where $y_j^{\text{ev}}$ and $y_i$ are labels. We compute the KNN-SVs of all examples in $D_c$ w.r.t. the evaluation point with the utility function (2). The KNN utility function over a subset $S \subset D_c$ measures the likelihood of correct classifications:

$$v_{j,\text{KNN}}(S) = \frac{1}{K} \sum_{k=1}^{\min(K,|S|)} \mathbb{1}[y_{\alpha_k(S)} = y_j^{\text{ev}}] \tag{2}$$

where $\alpha_k(S)$ is the index of the $k$th closest sample (from $\mathbf{x}_j^{\text{ev}}$) in $S$ based on some distance metric. Each sample $i$ is assigned a KNN-SV — $s_j(i)$ — that represents the average marginal contribution of the instance to the utility. Due to the additivity of SV, we obtain the KNN-SV of a candidate sample w.r.t. the evaluation set ($D_e = \{(\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}})\}_{j=1}^{N_e}$) by taking the average: $s_{\text{avg}}(i) = {}^1/_{N_e} \sum_{j=1}^{N_e} s_j(i)$.

(3) and (4) show how to recursively compute the exact KNN-SVs of samples in $D_c$ w.r.t. $(\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}}) \in D_e$ starting from $\mathbf{x}_{\alpha_{N_c}}$ (the farthest point from $\mathbf{x}_j^{\text{ev}}$) (Jia et al. 2019a):

$$s_j(\alpha_{N_c}) = \frac{\mathbb{1}[y_{\alpha_{N_c}} = y_j^{\text{ev}}]}{N_c} \tag{3}$$

$$s_j(\alpha_m) = s_j(\alpha_{m+1}) + \frac{\mathbb{1}[y_{\alpha_m} = y_j^{\text{ev}}] - \mathbb{1}[y_{\alpha_{m+1}} = y_j^{\text{ev}}]}{K} \frac{\min(K,m)}{m} \tag{4}$$

Here, $s_j(\alpha_m)$ is the KNN-SV of the $m$th closest candidate sample from $\mathbf{x}_j^{\text{ev}}$. Note that the dependency on the utility $v$ is suppressed as $v_{\text{KNN}}$ is always used. We refer readers to (Jia et al. 2019a) for detailed derivation of these results.

## 4 Adversarial Shapley Value Experience Replay (ASER)

We have now affirmatively answered how to value data in the embedded space in terms of its contribution to accurate classification by leveraging the efficient KNN-SV computation. Equipped with this powerful global data valuation algorithm, we now present our novel ER method dubbed Adversarial Shapley value ER (ASER) that leverages the SV for both *MemoryRetrieval* and *MemoryUpdate*.

A key insight with our ASER approach for *MemoryRetrieval* is that we need to balance the competing needs at the crux of CL, i.e., we need to retrieve memory samples for replay that prevent forgetting while also finding samples that maximally interfere with the incoming batch $B_n$ to ensure plasticity in learning. This leads us not only to leverage a cooperative notion of the SV (where higher SV is better) as it relates to $\mathcal{M}$ but also an adversarial notion of the SV as it relates to $B_n$ (where lower – and, in fact, negative – SVs indicate interference). In addition ASER also adopts a cooperative SV approach to the *MemoryUpdate* process.

Formally, we can view a neural network classifier ($f$) as two separate parts: a feature extractor ($f_{\text{ext}} : \mathbb{R}^d \mapsto \mathbb{R}^h$) and a fully connected neural classifier ($f_{\text{cl}} : \mathbb{R}^h \mapsto \mathbb{R}^C$), where $h$ is the dimensionality of the latent space $\mathcal{X}^l$. We implicitly define a KNN classifier and use the Euclidean distance in $\mathcal{X}^l$. Then, by (3)-(4), we can compute the KNN-SVs of candidate samples w.r.t. evaluation samples.

As previously noted, ER's performance depends on deciding what to store in memory (i.e., *MemoryUpdate*) and what to replay from memory (i.e., *MemoryRetrieval*). One key desiderata is that we want samples in $\mathcal{M}$ as well as $B_n$ to be well-separated by $f_{\text{ext}}$ in the latent space. To this end, we target two types of samples in $\mathcal{M}$ for retrieval: those near the samples in $B_n$ but have different labels (Type 1); those that are representative of samples in the memory (Type 2). Training with samples in Type 1 encourages the model to learn to differentiate current classes from previously seen classes. Samples in Type 2 help retain latent decision boundaries for previously observed classes.

We ground our intuition as to how samples interfere and cluster with each other in the latent space based on two properties of the KNN-SV. Given a candidate sample $i \in D_c$ and an evaluation set $D_e$, the KNN-SV of the point $i$ w.r.t. an evaluation point $j \in D_e$, i.e. $s_j(i)$, satisfies the following (see Appendix[1] A for proof):

- **Property 1.** $s_j(i) > 0$ if and only if $y_i = y_j^{\text{ev}}$. Also, $s_j(i) = 0$ only when $S = \{i' | y_{i'} = y_j^{\text{ev}}, \forall i' \in \{i+1, \ldots, N_c\}\} = \emptyset$.
- **Property 2.** $|s_j(m)|$ is a non-increasing function of $m$ for $m$ such that $y_m = y_j^{\text{ev}}$. Similarly, $|s_j(n)|$ is a non-increasing function of $n$ for $n$ such that $y_n \neq y_j^{\text{ev}}$. And for $m \geq K$, $|s_j(m)| - |s_j(m')| > 0$ holds for $m < m'$, where $m'$ is the smallest index with $\mathbb{1}(y_m = y_j^{\text{ev}}) = \mathbb{1}(y_{m'} = y_j^{\text{ev}})$, if there exists $l \in (m, m')$ such that $\mathbb{1}(y_l = y_j^{\text{ev}}) \neq \mathbb{1}(y_m = y_j^{\text{ev}})$. In other words, as $i$ gets closer to the evaluation point $j$, $|s_j(i)|$ cannot decrease for points with the same $\mathbb{1}(y_i = y_j^{\text{ev}})$, and for $i \geq K$, it can only increase when there exist more than one differently labeled points.

The first property states that a candidate sample $i$ has a *positive KNN-SV* if it has the same label as the evaluation point being considered (*cooperative*); the sample will have a *negative KNN-SV* if its label is different than the evaluation point (*adversarial*). By combining both properties, we note:

[1] Please find the appendix in our extended version on arXiv. Link: https://arxiv.org/abs/2009.00093

If $s_j(i)$ is large, the candidate $i$ is close to the evaluation point $j$ in the latent space ($\mathcal{X}^l$) and has the same label ($y_i = y_j^{\text{ev}}$). On the other hand, if $s_j(i)$ is a negative value of large magnitude, then $i$ is close to $j$, yet has a different label ($y_i \neq y_j^{\text{ev}}$). Thus, we conjecture that a good data candidate $i$ has *high positive SV* for memory $\mathcal{M}$ and *negative SV with large magnitude* for the current input task $B_n$.

When we consider the whole evaluation set, we take the mean $s_{D_e}(i) = {}^1\!/_{|D_e|} \cdot \sum_{j \in D_e} s_j(i)$, and the above analysis still holds in average. Therefore, by examining the KNN-SVs of candidate samples, we can get a sense of how they are distributed with respect to the evaluation set in $\mathcal{X}^l$. Then, we define the **adversarial SV (ASV)** that encodes the Type 1 & 2 criteria

$$\mathbf{ASV}(i) = \max_{j \in S_{\text{sub}}} s_j(i) - \min_{k \in B_n} s_k(i), \tag{5}$$

as well as a "softer" mean variation $\mathbf{ASV}_\mu$

$$\mathbf{ASV}_\mu(i) = \frac{1}{|S_{\text{sub}}|} \sum_{j \in S_{\text{sub}}} s_j(i) - \frac{1}{b} \sum_{k \in B_n} s_k(i), \tag{6}$$

where $i \in \mathcal{M} \setminus S_{\text{sub}}$ and $S_{\text{sub}}$ is constructed by subsampling some number of examples from $\mathcal{M}$ such that it is balanced in terms of the number of examples from each class. This prevents us from omitting any latent decision boundaries of classes in the memory. Note that $S_{\text{sub}}$ is used as the evaluation set in the first term, whereas the input batch $B_n$ forms the evaluation set in the latter term. The candidate set is $\bar{\mathcal{M}} = \mathcal{M} \setminus S_{\text{sub}}$, and we retrieve samples of size $b_{\mathcal{M}}$ from the set that have the highest **ASV**s (Algorithm 2). We denote our ER method using the score **ASV** (5) as ASER, while ASER$_\mu$ uses $\mathbf{ASV}_\mu$ (6) instead. For computational efficiency, we randomly subsample $N_c$ candidates from $\bar{\mathcal{M}}$.

**Algorithm 2:** ASER *MemoryRetrieval*

**Input :** Memory batch size $b_{\mathcal{M}}$
Input batch $B_n$; Candidate size $N_c$;
Subsample size $N_{\text{sub}}$;
Feature extractor $f_{\text{ext}}$

$S_{\text{sub}} \overset{N_{\text{sub}}}{\sim} \mathcal{M}$ `// get evaluation set`
$D_c \overset{N_c}{\sim} \mathcal{M} \setminus S_{\text{sub}}$ `// get candidate set`
`/* get latent embeddings */`
$L_{B_n}, L_{S_{\text{sub}}}, L_{D_c} \leftarrow f_{\text{ext}}(B_n), f_{\text{ext}}(S_{\text{sub}}), f_{\text{ext}}(D_c)$
**for** $i \in D_c$ **do**
  **for** $j \in S_{sub}$ **do**
    $s_j(i) \leftarrow$ KNN-SV($L_{D_c}, L_{S_{\text{sub}}}$) as per (3), (4)
  **for** $k \in B_n$ **do**
    $s_k(i) \leftarrow$ KNN-SV($L_{D_c}, L_{B_n}$) as per (3), (4)
  $score(i) \leftarrow \mathbf{ASV}(i)$ as per (5) or (6)
$B_{\mathcal{M}} \leftarrow b_{\mathcal{M}}$ samples with largest $score(\cdot)$
**return** $B_{\mathcal{M}}$

Note that both ASER methods do not greedily retrieve samples with the smallest distances to either $S_{\text{sub}}$ or $B_n$. This is because for a single evaluation point $j$, $s_j(\alpha_m) = s_j(\alpha_{m+1})$ when $y_{\alpha_m} = y_{\alpha_{m+1}}$. So, a few points can have the same score even if some of them are farther from the evaluation point. This is in contrast to a pure distance-based score where the closest point gets the highest score. In Appendix[1] B, we show that our method outperforms pure distance-based methods, proving the effectiveness of the global way in which the SV scores candidate data based on the KNN perspective.

We summarize our method in Algorithm 2, and compare it with other state-of-the-art ER methods on multiple challenging CL benchmarks in Section 5.

**Memory Update Based on KNN-SV** For *MemoryUpdate*, we find that samples with high KNN-SV promote clustering effect in the latent space. Therefore, they are useful to store in the memory, which aligns with the original meaning of the SV. More concretely, we subsample $S_{\text{sub}} \sim \mathcal{M}$ and compute ${}^1\!/_{|S_{\text{sub}}|} \sum_{j \in S_{\text{sub}}} s_j(i)$ for $i \in \bar{\mathcal{M}} \cup B_n$. Then, we replace samples in $\bar{\mathcal{M}}$ having smaller average KNN-SVs than samples in $B_n$ with the input batch samples.

We use KNN-SV *MemoryUpdate* for ASER throughout experiments in Section 5, while the ablation analysis of different variations with random *MemoryUpdate* or random *MemoryRetrieval* (both random retrieval and update reduces to ER) is presented in Appendix[1] C. Note that ASER with KNN-SV *MemoryUpdate* performs competitively or better than the variations, underscoring the importance of SV-based methods for both *MemoryUpdate* and *MemoryRetrieval*.

## 5 Experiments

To test the efficacy of ASER and its variant ASER$_\mu$, we evaluate their performance by comparing them with several state-of-the-art CL baselines. We begin by reviewing the benchmark datasets, baselines we compared against and our experiment setting. We then report and analyze the result to validate our approach.

### Datasets

**Split CIFAR-10** splits the CIFAR-10 dataset (Krizhevsky 2009) into 5 different tasks with non-overlapping classes and 2 classes in each task, similarly as in (Aljundi et al. 2019a).
**Split CIFAR-100** is constructed by splitting the CIFAR-100 dataset (Krizhevsky 2009) into 10 disjoint tasks, and each task has 10 classes.
**Split miniImagenet** consists of splitting the miniImageNet dataset (Vinyals et al. 2016) into 10 disjoint tasks, where each task contains 10 classes

The detail of datasets, including the general information of each dataset, class composition and the number of samples in training, validation and test sets of each task is presented in Appendix[1] D.

### Baselines

We compare our proposed ASER against several state-of-the-art continual learning algorithms:

| Method | M=1k | M=2k | M=5k | M=1k | M=2k | M=5k | M=0.2k | M=0.5k | M=1k |
|---|---|---|---|---|---|---|---|---|---|
| iid online | $14.7 \pm 0.6$ | $14.7 \pm 0.6$ | $14.7 \pm 0.6$ | $20.5 \pm 0.4$ | $20.5 \pm 0.4$ | $20.5 \pm 0.4$ | $62.9 \pm 1.5$ | $62.9 \pm 1.5$ | $62.9 \pm 1.5$ |
| iid offline | $42.4 \pm 0.4$ | $42.4 \pm 0.4$ | $42.4 \pm 0.4$ | $47.4 \pm 0.3$ | $47.4 \pm 0.3$ | $47.4 \pm 0.3$ | $79.7 \pm 0.4$ | $79.7 \pm 0.4$ | $79.7 \pm 0.4$ |
| AGEM | $7.0 \pm 0.4$ | $7.1 \pm 0.5$ | $6.9 \pm 0.7$ | $9.5 \pm 0.4$ | $9.3 \pm 0.4$ | $9.7 \pm 0.3$ | $22.7 \pm 1.8$ | $22.7 \pm 1.9$ | $22.6 \pm 0.7$ |
| ER | $8.7 \pm 0.4$ | $11.8 \pm 0.9$ | $16.5 \pm 0.9$ | $11.2 \pm 0.4$ | $14.6 \pm 0.4$ | $20.1 \pm 0.8$ | $26.4 \pm 1.0$ | $32.2 \pm 1.4$ | $38.4 \pm 1.7$ |
| EWC | $3.1 \pm 0.3$ | $3.1 \pm 0.3$ | $3.1 \pm 0.3$ | $4.8 \pm 0.2$ | $4.8 \pm 0.2$ | $4.8 \pm 0.2$ | $17.9 \pm 0.3$ | $17.9 \pm 0.3$ | $17.9 \pm 0.3$ |
| fine-tune | $4.3 \pm 0.2$ | $4.3 \pm 0.2$ | $4.3 \pm 0.2$ | $5.9 \pm 0.2$ | $5.9 \pm 0.2$ | $5.9 \pm 0.2$ | $17.9 \pm 0.4$ | $17.9 \pm 0.4$ | $17.9 \pm 0.4$ |
| GSS | $7.5 \pm 0.5$ | $10.7 \pm 0.8$ | $12.5 \pm 0.4$ | $9.3 \pm 0.2$ | $10.9 \pm 0.3$ | $15.9 \pm 0.4$ | $26.9 \pm 1.2$ | $30.7 \pm 1.2$ | $40.1 \pm 1.4$ |
| MIR | $8.1 \pm 0.3$ | $11.2 \pm 0.7$ | $15.9 \pm 1.6$ | $11.2 \pm 0.3$ | $14.1 \pm 0.2$ | $21.2 \pm 0.6$ | $28.3 \pm 1.6$ | $35.6 \pm 1.2$ | $42.4 \pm 1.5$ |
| ASER | $\mathbf{11.7 \pm 0.7}$ | $\mathbf{14.4 \pm 0.4}$ | $\mathbf{18.2 \pm 0.7}$ | $12.3 \pm 0.4$ | $14.7 \pm 0.7$ | $20.0 \pm 0.6$ | $27.8 \pm 1.0$ | $36.2 \pm 1.1$ | $43.1 \pm 1.2$ |
| ASER$_\mu$ | $\mathbf{12.2 \pm 0.8}$ | $\mathbf{14.8 \pm 1.1}$ | $\mathbf{18.2 \pm 1.1}$ | $\mathbf{14.0 \pm 0.4}$ | $\mathbf{17.2 \pm 0.5}$ | $21.7 \pm 0.5$ | $26.4 \pm 1.5$ | $36.3 \pm 1.2$ | $43.5 \pm 1.4$ |
| | (a) Mini-ImageNet | | | (b) CIFAR-100 | | | (c) CIFAR-10 | | |

Table 1: Average Accuracy (higher is better), M is the memory buffer size. All numbers are the average of 15 runs. ASER$_\mu$ has better performance when M is small and dataset is more complex.[2]

- **AGEM** (Chaudhry et al. 2019a): Averaged Gradient Episodic Memory, a memory-based method that utilizes the samples in the memory buffer to constrain the parameter updates.
- **ASER & ASER$_\mu$**: Our proposed methods. ASER scores samples in the memory with **ASV** in (5). ASER$_\mu$ uses the mean variation **ASV**$_\mu$ in (6).
- **ER** (Chaudhry et al. 2019b): Experience replay, a recent and successful rehearsal method with random sampling in *MemoryRetrieval* and reservoir sampling in *MemoryUpdate*.
- **EWC** (Kirkpatrick et al. 2017): Elastic Weight Consolidation, a prior-focused method that limits the update of parameters that were important to the past tasks, as measured by the Fisher information matrix.
- **GSS** (Aljundi et al. 2019b): Gradient-Based Sample Selection, a *MemoryUpdate* method that diversifies the gradients of the samples in the replay memory.
- **MIR** (Aljundi et al. 2019a): Maximally Interfered Retrieval, a *MemoryRetrieval* method that retrieves memory samples that suffer from an increase in loss given the estimated parameters update based on the current task.
- **iid-online & iid-offline**: iid-online trains the model with a single-pass through the same set of data, but each mini-batch is sampled iid from the training set. iid-offline trains the model over multiple epochs on the dataset with iid sampled mini-batch. We use 5 epochs for iid-offline in all the experiments as in (Aljundi et al. 2019a,b).
- **fine-tune**: As an important baseline in previous work (Aljundi et al. 2019a,b; Lee et al. 2020), it simply trains the model in the order the data is presented without any specific method for forgetting avoidance.

## Experiment Setting

**Single-head Evaluation** Most of the previous work in CL applied multi-head evaluation (Chaudhry et al. 2018) where a distinct output head is assigned for each task and the model utilizes the task identity to choose the corresponding output head during test time. But in many realistic scenarios, task identity is not available during test time, so the model should be able to classify labels from different tasks. As in (Aljundi et al. 2019a,b), we adopt the single-head evaluation setup where the model has one output head for all tasks and is required to classify all labels. Note that the setting we use – online and single-head evaluation – is more challenging than many other reported CL settings.

**Model** We use a reduced ResNet18, similar to (Chaudhry et al. 2019b; Lopez-Paz and Ranzato 2017), as the base model for all datasets, and the network is trained via cross-entropy loss with SGD optimizer and mini-batch size of 10. The size of the mini-batch retrieved from memory is also set to 10 irrespective of the size of the memory. More details of the experiment can be found in Appendix[1]E.

## Comparative Performance Evaluation

Table 1 and Table 2 show the average accuracy and average forgetting by the end of the data stream for Mini-ImageNet, CIFAR-100 and CIFAR-10. Based on the performance of iid-online and iid-offline, we verify that Mini-ImageNet and CIFAR-100 are more complex than CIFAR-10, even though three datasets have the same number of samples. Overall, ASER and ASER$_\mu$ show competitive or improved performance in three standard CL datasets. Especially, we observe that ASER$_\mu$ outperforms all the state-of-the-art baselines by significant margins in a more difficult setting where memory size is small and dataset is complex. Since the difficulty of the three datasets is different, comparing the absolute accuracy improvement may not be fair. Therefore, percentage improvement[3] is more appropriate here. Taking Mini-ImageNet as an example, ASER$_\mu$ improves the strongest baseline by 40.2% (M=1k), 25.4% (M=2k) and

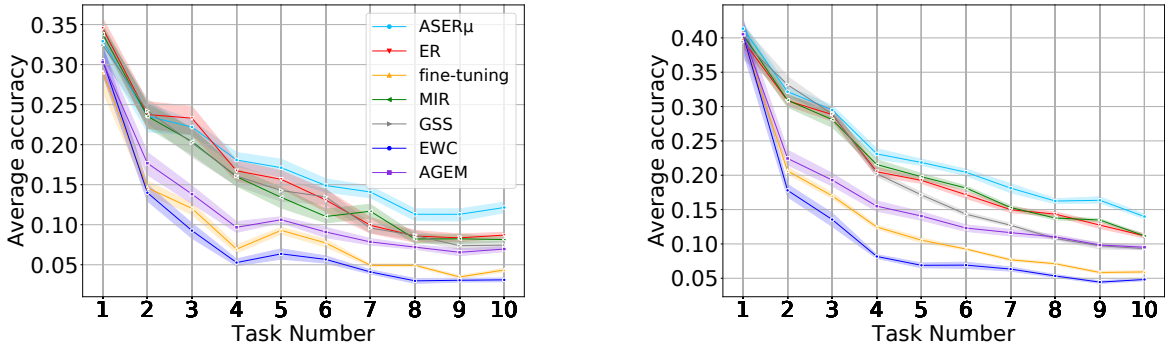
[2]The discrepancy of CIFAR-10 result for MIR between the original paper and this work is discussed in Appendix[1] F

[3]Percentage improvement is the ratio between absolute improvement and baseline performance. For example, in Mini-ImageNet(M=1k), ASER$_\mu$ improves MIR by $\frac{12.2-8.7}{8.7} = 40.2\%$

| Method | M=1k | M=2k | M=5k | M=1k | M=2k | M=5k | M=0.2k | M=0.5k | M=1k |
|---|---|---|---|---|---|---|---|---|---|
| AGEM | $29.3 \pm 0.9$ | $30.0 \pm 0.9$ | $29.9 \pm 0.8$ | $40.4 \pm 0.7$ | $39.7 \pm 0.8$ | $39.8 \pm 1.0$ | $36.1 \pm 3.8$ | $43.2 \pm 4.2$ | $48.1 \pm 3.0$ |
| ER | $29.7 \pm 1.3$ | $29.2 \pm 0.9$ | $26.6 \pm 1.1$ | $45.0 \pm 0.5$ | $40.5 \pm 0.8$ | $34.5 \pm 0.8$ | $72.8 \pm 1.7$ | $63.1 \pm 2.4$ | $55.8 \pm 2.6$ |
| EWC | $28.1 \pm 0.8$ | $28.1 \pm 0.8$ | $28.1 \pm 0.8$ | $39.1 \pm 1.2$ | $39.1 \pm 1.2$ | $39.1 \pm 1.2$ | $81.5 \pm 1.4$ | $81.5 \pm 1.4$ | $81.5 \pm 1.4$ |
| fine-tune | $35.6 \pm 0.9$ | $35.6 \pm 0.9$ | $35.6 \pm 0.9$ | $50.4 \pm 1.0$ | $50.4 \pm 1.0$ | $50.4 \pm 1.0$ | $81.7 \pm 0.7$ | $81.7 \pm 0.7$ | $81.7 \pm 0.7$ |
| GSS | $29.6 \pm 1.2$ | $27.4 \pm 1.1$ | $29.9 \pm 1.2$ | $46.9 \pm 0.7$ | $42.3 \pm 0.8$ | $39.2 \pm 0.9$ | $75.5 \pm 1.5$ | $65.9 \pm 1.6$ | $54.9 \pm 2.0$ |
| MIR | $29.7 \pm 1.0$ | $27.2 \pm 1.1$ | $26.2 \pm 1.4$ | $45.5 \pm 0.8$ | $40.4 \pm 0.6$ | $31.4 \pm 0.6$ | $67.0 \pm 2.6$ | $68.9 \pm 1.7$ | $47.7 \pm 2.9$ |
| ASER | $30.1 \pm 1.3$ | $\mathbf{24.7 \pm 1.0}$ | $\mathbf{20.9 \pm 1.2}$ | $50.1 \pm 0.6$ | $45.9 \pm 0.9$ | $36.7 \pm 0.8$ | $71.1 \pm 1.8$ | $\mathbf{59.1 \pm 1.5}$ | $50.4 \pm 1.5$ |
| ASER$_\mu$ | $28.0 \pm 1.3$ | $\mathbf{22.2 \pm 1.6}$ | $\mathbf{17.2 \pm 1.4}$ | $45.0 \pm 0.7$ | $\mathbf{38.6 \pm 0.6}$ | $30.3 \pm 0.5$ | $72.4 \pm 1.9$ | $\mathbf{58.8 \pm 1.4}$ | $47.9 \pm 1.6$ |

(a) Mini-ImageNet (b) CIFAR-100 (c) CIFAR-10

Table 2: Average Forgetting (lower is better). Memory buffer size is M. All numbers are the average of 15 runs.




(a) Mini-ImageNet (b) CIFAR-100

Figure 2: Average accuracy on observed tasks when M=1k. The shaded region represents the 95% confidence interval. ASER$_\mu$ outperforms other baselines especially when the model sees more classes (each task contains new classes).

10.3% (M=5k) in terms of percentage improvement. Moreover, as we can see in Figure 2, ASER$_\mu$ is consistently better than other baselines in both datasets. We also note that ASER$_\mu$ generally performs better than ASER. This is because if we use the ASV criterion as in (5), it has a higher chance that the value is affected by an outlier point in the *evaluation* set. So the $\mathbf{ASV}_\mu$ in (6) gives a more stable and accurate value in complicated datasets.

Another interesting observation is that ER has very competitive performances. Especially in more complex datasets, it surpasses GSS and performs similarly as MIR, which proves it to be a simple but powerful CL baseline. In addition, we find that for complex datasets, when memory size is larger than 5000 (10% of the training data), most of the replay-based methods (except for GSS) outperform the iid-online, a baseline that trains the model with a one-pass through the data but with iid-sampled mini-batch from the whole dataset. This means that storing a small number of training samples is crucial for combating forgetting as well as the learning of the current task in the online class-incremental setting.

We also verify claims from previous work (Lesort, Stoian, and Filliat 2019; Farquhar and Gal 2018; Aljundi et al. 2019a). EWC, a regularization-based method, not only is surpassed by all memory-based methods but also underperforms the fine-tuning baseline. Additionally, AGEM, a method that uses memory samples to constrain parameter updates, delivers worse performance compared with reply-based methods (ER, MIR, and GSS), especially when memory size increases.

Overall, by evaluating on three standard CL datasets and comparing to the state-of-the-art CL methods, we have shown the effectiveness of ASER and its variant ASER$_\mu$ in overcoming catastrophic forgetting, especially in more complex datasets and memory size is relatively small.

## 6 Conclusion

In this work, we proposed a novel ASER method that scores memory data samples according to their ability to preserve latent decision boundaries for previously observed classes while interfering with latent decision boundaries of current classes being learned. Overall, in the online class-incremental setting, we observed that ASER and its ASER$_\mu$ variant provide competitive or improved performance on a variety of datasets compared to state-of-the-art ER-based continual learning methods. We also remark that this work paves the way for a number of interesting research directions building on this work. Although our SV-based method has greatly improved the memory retrieval and update strategies, we may be able to do better than simply concatenating retrieved samples with the incoming batch. Hence, future work could focus on more sophisticated methods to utilize the retrieved samples. It would also be interesting to investigate alternate CL-specific utility function variations for SV.

## Acknowledgement

This research was supported by LG AI Research.

## References

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

Aljundi, R.; Belilovsky, E.; Tuytelaars, T.; Charlin, L.; Caccia, M.; Lin, M.; and Page-Caccia, L. 2019a. Online Continual Learning with Maximal Interfered Retrieval. In *Advances in Neural Information Processing Systems 32*, 11849–11860.

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

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

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2019a. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations*.

Chaudhry, A.; Rohrbach, M.; Elhoseiny, M.; Ajanthan, T.; Dokania, P. K.; Torr, P. H. S.; and Ranzato, M. 2019b. On Tiny Episodic Memories in Continual Learning.

Chrysakis, A.; and Moens, M.-F. 2020. Online Continual Learning from Imbalanced Data. *Proceedings of Machine Learning and Systems* 8303–8312.

De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383* .

Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; and Darrell, T. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, I–647–I–655. JMLR.org.

Farquhar, S.; and Gal, Y. 2018. Towards Robust Evaluations of Continual Learning.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P. H.; Kohli, P.; and Whiteson, S. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1146–1155. JMLR. org.

Ghorbani, A.; and Zou, J. 2019. Data Shapley: Equitable Valuation of Data for Machine Learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2242–2251. PMLR.

Jia, R.; Dao, D.; Wang, B.; Hubis, F. A.; Gurel, N. M.; Li, B.; Zhang, C.; Spanos, C.; and Song, D. 2019a. Efficient Task-Specific Data Valuation for Nearest Neighbor Algorithms. *Proc. VLDB Endow.* 12(11): 1610–1623. ISSN 2150-8097. doi:10.14778/3342263.3342637.

Jia, R.; Dao, D.; Wang, B.; Hubis, F. A.; Hynes, N.; Gürel, N. M.; Li, B.; Zhang, C.; Song, D.; and Spanos, C. J. 2019b. Towards Efficient Data Valuation Based on the Shapley Value. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, 1167–1176. PMLR.

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 of the United States of America* 114 13: 3521–3526.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto.

Lee, S.; Ha, J.; Zhang, D.; and Kim, G. 2020. A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning. In *International Conference on Learning Representations*.

Lesort, T.; Caselles-Dupré, H.; Ortiz, M. G.; Stoian, A.; and Filliat, D. 2018. Generative Models from the perspective of Continual Learning.

Lesort, T.; Stoian, A.; and Filliat, D. 2019. Regularization Shortcomings for Continual Learning. *arXiv preprint 1912.03049* .

Li, Z.; and Hoiem, D. 2016. Learning Without Forgetting. In *ECCV*, 614–629. Springer.

Lopez-Paz, D.; and Ranzato, M. A. 2017. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems 30*, 6467–6476.

Mallya, A.; and 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*, 7765–7773.

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

Min, R.; Stanley, D. A.; Yuan, Z.; Bonner, A.; and Zhang, Z. 2009. A Deep Non-Linear Feature Mapping for Large-Margin KNN Classification. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, 357–366. IEEE Computer Society. doi:10.1109/ICDM.2009.27.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113: 54 – 71.

Rannen, A.; Aljundi, R.; Blaschko, M. B.; and Tuytelaars, T. 2017. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 1320–1328.

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

Reimer, M.; Cases, I.; Ajemian, R.; Liu, M.; Rish, I.; Tu, Y.; and Tesauro, G. 2019. Learning to Learn without Forgetting vy Maximizing Transfer and Minimizing Interference. In *ICLR*.

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

Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T.; and Wayne, G. 2019. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 348–358.

Roth, A. E. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.

Shapley, L. S. 1953. A value for n-person games. *Contributions to the Theory of Games* 2(28): 307–317.

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

van der Maaten, L.; and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9: 2579–2605.

Vinyals, O.; Blundell, C.; Lillicrap, T.; kavukcuoglu, k.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29*, 3630–3638. Curran Associates, Inc.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; and Fu, Y. 2019. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 374–382.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2018. Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3987–3995. JMLR. org.

# Appendix

## A Properties of KNN Shapley Value

We prove the properties of KNN-SV presented in Section 4. Given a candidate sample $i = (\mathbf{x}_i, y_i) \in D_c$ and an evaluation point $j = (\mathbf{x}_j^{\text{ev}}, y_j^{\text{ev}})$, where $D_c$ is a candidate set with $|D_c| = N_c$, we denote the KNN-SV of the point $i$ w.r.t. the evaluation point $j$ as $s_j(i)$. For notational convenience, we assume that points in $D_c$ are sorted based on distances from the evaluation point in ascending order. In other words, $d(i,j) \leq d(i',j) \;\; \forall i < i'$ where $d(i,j)$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j^{\text{ev}}$.

**Property 1.** *$s_j(i) > 0$ if and only if $y_i = y_j^{ev}$. Also, $s_j(i) = 0$ only when $S = \{i' | y_{i'} = y_j^{ev},\ \forall i' \in \{i+1, \ldots, N_c\}\} = \emptyset$.*

*Proof.* Firstly, we prove $s_j(i) > 0$ *if and only if* $y_i = y_j^{\text{ev}}$ along with another statement, $|s_j(i)| < \frac{1}{i-1}$. The proof is by induction, starting from the base case when $i = N_c$. When $i = N_c$, $s_j(N_c) = \frac{\mathbb{1}(y_{N_c} = y_j^{\text{ev}})}{N_c}$ as per (3). Hence, $s_j(N_c) > 0$ holds iff $y_{N_c} = y_j^{\text{ev}}$. Additionally, we see that $|s_j(N_c)| \leq \frac{1}{N_c} < \frac{1}{N_c - 1}$. We can also verify the case for $i = N_c - 1$ using (4):

$$s_j(N_c - 1) = \begin{cases} \frac{1}{N_c} > 0 & \text{if } y_{N_c-1} = y_{N_c} = y_j^{\text{ev}} \\ \frac{1}{N_c - 1} > 0 & \text{if } y_{N_c-1} = y_j^{\text{ev}} \neq y_{N_c} \\ \frac{1}{N_c} - \frac{1}{N_c - 1} < 0 & \text{if } y_{N_c-1} \neq y_j^{\text{ev}} \text{ and } y_{N_c} = y_j^{\text{ev}} \\ 0 & \text{if } y_{N_c-1} \neq y_j^{\text{ev}} \text{ and } y_{N_c} \neq y_j^{\text{ev}} \end{cases} \tag{A.1}$$

So, we again note that $s_j(N_c - 1) > 0$ iff $y_{N_c-1} = y_j^{\text{ev}}$, and $|s_j(N_c - 1)| \leq \frac{1}{N_c - 1} < \frac{1}{N_c - 2}$ holds.

Now, assume for $i = m \geq K+1$, $s_j(m) > 0$ iff $y_m = y_j^{\text{ev}}$ and $|s_j(m)| < \frac{1}{m-1}$. Then, for $i = m - 1 \geq K$,

$$s_j(m-1) = s_j(m) + \frac{\mathbb{1}(y_{m-1} = y_j^{\text{ev}}) - \mathbb{1}(y_m = y_j^{\text{ev}})}{m-1} = \begin{cases} s_j(m) > 0 & \text{if } y_{m-1} = y_m = y_j^{\text{ev}}, \\ s_j(m) + \frac{1}{m-1} > 0 & \text{if } y_{m-1} = y_j^{\text{ev}} \neq y_m, \\ s_j(m) - \frac{1}{m-1} < 0 & \text{if } y_{m-1} \neq y_j^{\text{ev}} \text{ and } y_m = y_j^{\text{ev}}, \\ s_j(m) \leq 0 & \text{if } y_{m-1} \neq y_j^{\text{ev}} \text{ and } y_m \neq y_j^{\text{ev}} \end{cases} \tag{A.2}$$

Note that the second and the third cases in (A.2) hold because $|s_j(m)| < \frac{1}{m-1}$ by assumption. Additionally, it is straightforward to check that $|s_j(m-1)| < \frac{1}{m-2}$ holds for all cases. Hence, we have shown the statement holds for $m \geq K$.

For $m < K$, we firstly note that $|s_j(K)| \leq 1/K$ because $|s_j(K+1)| < 1/K$ and $s_j(K) = s_j(K+1) + \frac{\mathbb{1}(y_K = y_j^{\text{ev}}) - \mathbb{1}(y_{K+1} = y_j^{\text{ev}})}{K}$. Then, we see that the increment and the decrement, if any, are always $\frac{1}{K}$ for $m < K$. Therefore, $s_j(m) > 0$ iff $y_m = y_j^{\text{ev}}$. Furthermore, we have noted that the sign of $s_j(i)$ has to change iff $\mathbb{1}(y_i = y_j^{\text{ev}}) \neq \mathbb{1}(y_{i+1} = y_j^{\text{ev}})$ because $|s_j(i+1)| < 1/i$. This implies $s_j(i) = 0$ only when $S = \{i' | y_{i'} = y_j^{\text{ev}},\ \forall i' \in \{i+1, \ldots, N_c\}\} = \emptyset$, and we conclude the proof. □

**Property 2.** *$|s_j(m)|$ is a non-increasing function of $m$ for $m$ such that $y_m = y_j^{ev}$. Similarly, $|s_j(n)|$ is a non-increasing function of $n$ for $n$ such that $y_n \neq y_j^{ev}$. And for $m \geq K$, $|s_j(m)| - |s_j(m')| > 0$ holds for $m < m'$, where $m'$ is the smallest index with $\mathbb{1}(y_m = y_j^{ev}) = \mathbb{1}(y_{m'} = y_j^{ev})$, if there exists $l \in (m, m')$ such that $\mathbb{1}(y_l = y_j^{ev}) \neq \mathbb{1}(y_m = y_j^{ev})$. In other words, as $i$ gets closer to the evaluation point $j$, $|s_j(i)|$ cannot decrease for points with the same $\mathbb{1}(y_i = y_j^{ev})$, and for $i \geq K$, it can only increase when there exist more than one differently labeled points.*

*Proof.* We only show for $m$ such that $y_m = y_j^{\text{ev}}$ as it can be similarly done for $n$ with $y_n \neq y_j^{\text{ev}}$. If $y_l = y_j^{\text{ev}} \;\; \forall l \in (m, m']$, then it trivially holds that $s_j(m) = s_j(m')$. Now, assume that there exists one $l \in (m, m')$ such that $y_l \neq y_j^{\text{ev}}$ and that $l = m+1$. Then, we see that $|s_j(m)| - |s_j(m')| > 0$ holds. This is because

$$\begin{aligned} s_j(m) &= s_j(m+1) + \frac{\mathbb{1}(y_m = y_j^{\text{ev}}) - \mathbb{1}(y_{m+1} = y_j^{\text{ev}})}{m} \\ &= s_j(m+2) + \frac{\mathbb{1}(y_{m+1} = y_j^{\text{ev}}) - \mathbb{1}(y_{m+2} = y_j^{\text{ev}})}{m+1} + \frac{1}{m} \\ &= s_j(m+2) - \frac{1}{m+1} + \frac{1}{m} \\ &= s_j(m+2) + \left(\frac{1}{m} - \frac{1}{m+1}\right) \\ &> s_j(m+2) = s_j(m'), \;\; \forall m' \geq m+2 \end{aligned} \tag{A.3}$$

Then, we note that $s_j(m'') = s_j(m)$ for all $m'' \in (i, m]$ where $i < m$ is the largest index with $y_i \neq y_j^{\text{ev}}$ (if exists) or $i = 0$. This shows that $|s_j(m)| > |s_j(m')|$ holds for $m <$

$m'$ if there is a single $l \in (m, m')$ with $y_l \neq y_j^{\text{ev}}$. When there are multiple (possibly consecutive) points with $y_l \neq y_j^{\text{ev}}$, we can always select $\hat{m} \geq m$ such that there is only a single point (or several consecutive points) $l$ with $y_l \neq y_j^{\text{ev}}$, leading to $|s_j(\hat{m})| > |s_j(m')|$. By applying this multiple times, we get $|s_j(m)| > s_j(m')|$. □

# B Detailed Performance Evaluation

Following the definition of **ASV** and **ASV**$_\mu$, we can replace the Shapley value with distance (we use Euclidean as example) in (5) and (6). Specifically, we want to retrieve a point $i$ such that its distances from samples of the same label in $\mathcal{M}$ and its distances from input batch samples are both small. Concretely, the score for a candidate point $i$ is defined as follows:

$$\textbf{Dist}(i) = -\Big[\min_{j \in S_{\text{sub}}(i)} d(i,j) + \min_{k \in B_n} d(i,k)\Big], \tag{B.1}$$

as well as a "softer" mean variation **Dist**$_\mu$

$$\textbf{Dist}_\mu(i) = -\left[\frac{1}{|S_{\text{sub}}(i)|}\sum_{j \in S_{\text{sub}}(i)} d(i,j) + \frac{1}{b}\sum_{k \in B_n} d(i,k)\right]. \tag{B.2}$$

Here, $i \in \mathcal{M} \setminus S_{\text{sub}}$ and $S_{\text{sub}}$ is defined as in Section 4 and $S_{\text{sub}}(i) = \{i' | i' \in S_{\text{sub}} \ \& \ y_{i'} = y_i\}$. $d(i,j)$ is the Euclidean distance between the candidate point $i$ and an evaluation point $j$ in the latent space. Finally, we simply replace the score in Algorithm 2 (line 9) with either one of the above scores.

## Detailed Result Tables

In addition to the algorithms listed in Section 5, in Table A.1 and Table A.2, we include more baselines for comparison:

- **AGEM** (Chaudhry et al. 2019a): Averaged Gradient Episodic Memory, a memory-based method that utilizes the samples in the memory buffer to constrain the parameter updates.
- **ASER & ASER**$_\mu$: Our proposed methods. ASER scores samples in the memory with **ASV** in (5). ASER$_\mu$ uses the mean variation **ASV**$_\mu$ in (6).
- **Dist & Dist**$_\mu$: The Euclidean variants of ASER & ASER$_\mu$ that replace Shapley value with Euclidean distance, as described above.
- **ER** (Chaudhry et al. 2019b): Experience replay, a recent and successful rehearsal method with random sampling in *MemoryRetrieval* and reservoir sampling in *MemoryUpdate*.
- **EWC** (Kirkpatrick et al. 2017): Elastic Weight Consolidation, a prior-focused method that limits the update of parameters that were important to the past tasks, as measured by the Fisher information matrix.
- **GSS** (Aljundi et al. 2019b): Gradient-Based Sample Selection, a *MemoryUpdate* method that diversifies the gradients of the samples in the replay memory.
- **MIR** (Aljundi et al. 2019a): Maximally Interfered Retrieval, a *MemoryRetrieval* method that retrieves memory samples that suffer from an increase in loss given the estimated parameters update based on the current task.
- **iid-online & iid-offline**: iid-online trains the model with a single-pass through the same set of data, but each mini-batch is sampled iid from the training set. iid-offline trains the model over multiple epochs on the dataset with iid sampled mini-batch. We use 5 epochs for iid-offline in all the experiments as in (Aljundi et al. 2019a,b).
- **fine-tune**: As an important baseline in previous work (Aljundi et al. 2019a,b; Lee et al. 2020), it simply trains the model in the order the data is presented without any specific method for forgetting avoidance.

**Average Accuracy and Forgetting** As we can see from Table A.1 and Table A.2, ASER and ASER$_\mu$ outperform Dist and Dist$_\mu$. The reason may be that both ASER methods do not greedily retrieve samples with the smallest distances to either $S_{\text{sub}}$ (sub-sample from $\mathcal{M}$) or $B_n$(incoming mini-batch). This is because for a single evaluation point $j$, $s_j(\alpha_m) = s_j(\alpha_{m+1})$ when $y_{\alpha_m} = y_{\alpha_{m+1}}$. So, a few points can have the same score even if some of them are farther from the evaluation point.

We also verify some claims from previous work (Lesort, Stoian, and Filliat 2019; Farquhar and Gal 2018; Aljundi et al. 2019a). EWC, a prior-focused method, not only is surpassed by all memory-based methods but also underperforms the fine-tuning baseline. Additionally, AGEM, a method that uses memory samples to constrain parameter updates, delivers worse performance compared with reply-based methods (ER, MIR, and GSS), especially when memory size increases.

**Training Time** EWC, ER and AGEM have similar training time and their training times are almost twice as the finetune baseline. Since MIR, GSS and our proposed ASER need to perform additional calculation during *MemoryRetrieval* and *MemoryUpdate*, the training times are longer than the methods mentioned above. ASER takes longer than MIR because MIR only has additional computation in *MemoryRetrieval* but ASER carefully selects samples in both *MemoryRetrieval* and *MemoryUpdate*. Compared with GSS, ASER is more computationally efficient. Figure.B.1

# C Ablation Studies

In ASER & ASER$_\mu$, we use ASV and ASV$_\mu$ for scoring samples for *MemoryRetrieval*, while KNN-SV is used for scoring samples for *MemoryUpdate*. In this part, we examine several ablations to understand contributions of each component in ASER methods. In addition to ASER, ASER$_\mu$ and ER as in Section 5, we compare 3 more variations:

- **SV-upd**: Use KNN-SV *MemoryUpdate* as described in Section 4 while randomly retrieving samples from the memory for replay.
- **ASV-ret**: Use (5) scoring function for *MemoryRetrieval* while using reservoir sampling for *MemoryUpdate*.

| Method | M=1k | M=2k | M=5k |
|---|---|---|---|
| iid online | $14.7 \pm 0.6$ | $14.7 \pm 0.6$ | $14.7 \pm 0.6$ |
| iid offline | $42.4 \pm 0.4$ | $42.4 \pm 0.4$ | $42.4 \pm 0.4$ |
| AGEM | $7.0 \pm 0.4$ | $7.1 \pm 0.5$ | $6.9 \pm 0.7$ |
| ER | $8.7 \pm 0.4$ | $11.8 \pm 0.9$ | $16.5 \pm 0.9$ |
| EWC | $3.1 \pm 0.3$ | $3.1 \pm 0.3$ | $3.1 \pm 0.3$ |
| fine-tune | $4.3 \pm 0.2$ | $4.3 \pm 0.2$ | $4.3 \pm 0.2$ |
| GSS | $7.5 \pm 0.5$ | $10.7 \pm 0.8$ | $12.5 \pm 0.4$ |
| MIR | $8.1 \pm 0.3$ | $11.2 \pm 0.7$ | $15.9 \pm 1.6$ |
| ASER | $\mathbf{11.7 \pm 0.8}$ | $\mathbf{14.4 \pm 0.4}$ | $\mathbf{18.2 \pm 0.7}$ |
| ASER$_\mu$ | $\mathbf{12.2 \pm 0.8}$ | $\mathbf{14.8 \pm 1.1}$ | $\mathbf{18.2 \pm 1.1}$ |
| Dist | $7.9 \pm 0.5$ | $10.4 \pm 0.7$ | $15.5 \pm 0.9$ |
| Dist$_\mu$ | $8.1 \pm 0.5$ | $9.1 \pm 0.7$ | $14.9 \pm 0.9$ |

(a) Mini-ImageNet

| M=1k | M=2k | M=5k |
|---|---|---|
| $20.5 \pm 0.4$ | $20.5 \pm 0.4$ | $20.5 \pm 0.4$ |
| $47.4 \pm 0.3$ | $47.4 \pm 0.3$ | $47.4 \pm 0.3$ |
| $9.5 \pm 0.4$ | $9.3 \pm 0.4$ | $9.7 \pm 0.3$ |
| $11.2 \pm 0.4$ | $14.6 \pm 0.4$ | $20.1 \pm 0.8$ |
| $4.8 \pm 0.2$ | $4.8 \pm 0.2$ | $4.8 \pm 0.2$ |
| $5.9 \pm 0.2$ | $5.9 \pm 0.2$ | $5.9 \pm 0.2$ |
| $9.3 \pm 0.2$ | $10.9 \pm 0.3$ | $15.9 \pm 0.4$ |
| $11.2 \pm 0.3$ | $14.1 \pm 0.2$ | $21.2 \pm 0.6$ |
| $12.3 \pm 0.4$ | $14.7 \pm 0.7$ | $20.0 \pm 0.6$ |
| $\mathbf{14.0 \pm 0.4}$ | $\mathbf{17.2 \pm 0.5}$ | $21.7 \pm 0.5$ |
| $10.3 \pm .3$ | $12.4 \pm 0.5$ | $16.7 \pm 0.6$ |
| $10.5 \pm 0.2$ | $13.4 \pm 0.4$ | $17.2 \pm 0.8$ |

(b) CIFAR-100

| M=0.2k | M=0.5k | M=1k |
|---|---|---|
| $62.9 \pm 1.5$ | $62.9 \pm 1.5$ | $62.9 \pm 1.5$ |
| $79.7 \pm 0.4$ | $79.7 \pm 0.4$ | $79.7 \pm 0.4$ |
| $22.7 \pm 1.8$ | $22.7 \pm 1.9$ | $22.6 \pm 0.7$ |
| $26.4 \pm 1.0$ | $32.2 \pm 1.4$ | $38.4 \pm 1.7$ |
| $17.9 \pm 0.3$ | $17.9 \pm 0.3$ | $17.9 \pm 0.3$ |
| $17.9 \pm 0.4$ | $17.9 \pm 0.4$ | $17.9 \pm 0.4$ |
| $26.9 \pm 1.2$ | $30.7 \pm 1.2$ | $40.1 \pm 1.4$ |
| $28.3 \pm 1.6$ | $35.6 \pm 1.2$ | $42.4 \pm 1.5$ |
| $27.8 \pm 1.0$ | $36.2 \pm 1.1$ | $43.1 \pm 1.2$ |
| $26.4 \pm 1.5$ | $36.3 \pm 1.2$ | $43.5 \pm 1.4$ |
| $22.9 \pm 0.9$ | $31.6 \pm 1.7$ | $38.0 \pm 2.3$ |
| $23.4 \pm 1.0$ | $29.7 \pm 1.2$ | $35.6 \pm 1.3$ |

(c) CIFAR-10

Table A.1: Average Accuracy(higher is better). Memory buffer size M.

| Method | M=1k | M=2k | M=5k |
|---|---|---|---|
| AGEM | $29.3 \pm 0.9$ | $30.0 \pm 0.9$ | $29.9 \pm 0.8$ |
| ER | $29.7 \pm 1.3$ | $29.2 \pm 0.9$ | $26.6 \pm 1.1$ |
| EWC | $28.1 \pm 0.8$ | $28.1 \pm 0.8$ | $28.1 \pm 0.8$ |
| fine-tune | $35.6 \pm 0.9$ | $35.6 \pm 0.9$ | $35.6 \pm 0.9$ |
| GSS | $29.6 \pm 1.2$ | $27.4 \pm 1.1$ | $29.9 \pm 1.2$ |
| MIR | $29.7 \pm 1.0$ | $27.2 \pm 1.1$ | $26.2 \pm 1.4$ |
| ASER | $30.1 \pm 1.3$ | $24.7 \pm 1.0$ | $20.9 \pm 1.2$ |
| ASER$_\mu$ | $28.0 \pm 1.3$ | $\mathbf{22.2 \pm 1.6}$ | $\mathbf{17.2 \pm 1.4}$ |
| Dist | $30.4 \pm 1.1$ | $25.5 \pm 1.7$ | $26.3 \pm 1.2$ |
| Dist$_\mu$ | $29.7 \pm 1.1$ | $29.8 \pm 1.1$ | $27.1 \pm 1.4$ |

(a) Mini-ImageNet

| M=1k | M=2k | M=5k |
|---|---|---|
| $40.4 \pm 0.7$ | $39.7 \pm 0.8$ | $39.8 \pm 1.0$ |
| $45.0 \pm 0.5$ | $40.5 \pm 0.8$ | $34.5 \pm 0.8$ |
| $39.1 \pm 1.2$ | $39.1 \pm 1.2$ | $39.1 \pm 1.2$ |
| $50.4 \pm 1.0$ | $50.4 \pm 1.0$ | $50.4 \pm 1.0$ |
| $46.9 \pm 0.7$ | $42.3 \pm 0.8$ | $39.2 \pm 0.9$ |
| $45.5 \pm 0.8$ | $40.4 \pm 0.6$ | $31.4 \pm 0.6$ |
| $50.1 \pm 0.6$ | $45.9 \pm 0.9$ | $36.7 \pm 0.8$ |
| $45.0 \pm 0.7$ | $\mathbf{38.6 \pm 0.6}$ | $\mathbf{30.3 \pm 0.5}$ |
| $48.1 \pm 0.8$ | $43.8 \pm 0.5$ | $39.5 \pm 0.8$ |
| $48.0 \pm 0.6$ | $43.5 \pm 0.6$ | $39.9 \pm 0.8$ |

(b) CIFAR-100

| M=0.2k | M=0.5k | M=1k |
|---|---|---|
| $36.1 \pm 3.8$ | $43.2 \pm 4.2$ | $48.1 \pm 3.0$ |
| $72.8 \pm 1.7$ | $63.1 \pm 2.4$ | $55.8 \pm 2.6$ |
| $81.5 \pm 1.4$ | $81.5 \pm 1.4$ | $81.5 \pm 1.4$ |
| $81.7 \pm 0.7$ | $81.7 \pm 0.7$ | $81.7 \pm 0.7$ |
| $75.5 \pm 1.5$ | $65.9 \pm 1.6$ | $54.9 \pm 2.0$ |
| $\mathbf{67.0 \pm 2.6}$ | $68.9 \pm 1.7$ | $\mathbf{47.7 \pm 2.9}$ |
| $71.1 \pm 1.8$ | $59.1 \pm 1.5$ | $50.4 \pm 1.5$ |
| $72.4 \pm 1.9$ | $58.8 \pm 1.4$ | $47.9 \pm 1.6$ |
| $76.4 \pm 2.3$ | $\mathbf{63.7 \pm 2.3}$ | $53.6 \pm 3.8$ |
| $77.6 \pm 1.6$ | $68.6 \pm 2.0$ | $58.8 \pm 2.7$ |

(c) CIFAR-10

Table A.2: Average Forgetting (lower is better). Memory buffer size M.

| Method | M=1k | M=2k | M=5k |
|---|---|---|---|
| ER | $8.7 \pm 0.4$ | $11.8 \pm 0.9$ | $16.5 \pm 0.9$ |
| SV-upd | $\mathbf{13.4 \pm 0.8}$ | $\mathbf{15.5 \pm 0.7}$ | $\mathbf{18.4 \pm 0.4}$ |
| ASV-ret | $6.9 \pm 0.4$ | $9.9 \pm 0.9$ | $16.2 \pm 0.8$ |
| ASV$_\mu$-ret | $7.4 \pm 0.6$ | $10.0 \pm 1.0$ | $17.1 \pm 0.9$ |
| ASER | $11.7 \pm 0.8$ | $14.4 \pm 0.4$ | $\mathbf{18.2 \pm 0.7}$ |
| ASER$_\mu$ | $12.2 \pm 0.8$ | $\mathbf{14.8 \pm 1.1}$ | $\mathbf{18.2 \pm 1.1}$ |

(a) Mini-ImageNet

| M=1k | M=2k | M=5k |
|---|---|---|
| $11.2 \pm 0.4$ | $14.6 \pm 0.4$ | $20.1 \pm 0.8$ |
| $\mathbf{14.0 \pm 0.6}$ | $\mathbf{17.2 \pm 0.4}$ | $20.9 \pm 0.6$ |
| $10.1 \pm 0.3$ | $13.9 \pm 0.3$ | $20.3 \pm 0.3$ |
| $10.8 \pm 0.3$ | $14.8 \pm 0.4$ | $\mathbf{21.7 \pm 0.3}$ |
| $12.3 \pm 0.4$ | $14.7 \pm 0.7$ | $20.0 \pm 0.6$ |
| $\mathbf{14.0 \pm 0.4}$ | $\mathbf{17.2 \pm 0.5}$ | $\mathbf{21.7 \pm 0.5}$ |

(b) CIFAR-100

| M=0.2k | M=0.5k | M=1k |
|---|---|---|
| $26.4 \pm 1.0$ | $32.2 \pm 1.4$ | $38.4 \pm 1.7$ |
| $25.9 \pm 0.6$ | $33.7 \pm 1.4$ | $41.2 \pm 1.3$ |
| $26.6 \pm 1.0$ | $34.7 \pm 1.1$ | $39.3 \pm 1.6$ |
| $25.8 \pm 1.0$ | $35.7 \pm 1.4$ | $40.2 \pm 1.0$ |
| $\mathbf{27.8 \pm 1.0}$ | $36.2 \pm 1.1$ | $\mathbf{43.1 \pm 1.2}$ |
| $26.4 \pm 1.5$ | $36.3 \pm 1.2$ | $\mathbf{43.5 \pm 1.4}$ |

(c) CIFAR-10

Table C.1: Ablation analysis. Average Accuracy (higher is better). Memory buffer size M.

| Method | M=1k | M=2k | M=5k |
|---|---|---|---|
| ER | $29.7 \pm 1.3$ | $29.2 \pm 0.9$ | $26.6 \pm 1.1$ |
| SV-upd | $29.2 \pm 1.1$ | $26.4 \pm 1.5$ | $24.6 \pm 1.3$ |
| ASV-ret | $\mathbf{27.6 \pm 1.2}$ | $27.7 \pm 1.3$ | $25.5 \pm 1.2$ |
| ASV$_\mu$-ret | $\mathbf{28.3 \pm 1.3}$ | $27.4 \pm 1.2$ | $23.2 \pm 1.6$ |
| ASER | $30.1 \pm 1.3$ | $24.7 \pm 1.0$ | $20.9 \pm 1.2$ |
| ASER$_\mu$ | $\mathbf{28.0 \pm 1.3}$ | $\mathbf{22.2 \pm 1.6}$ | $\mathbf{17.2 \pm 1.4}$ |

(a) Mini-ImageNet

| M=1k | M=2k | M=5k |
|---|---|---|
| $45.5 \pm 0.5$ | $40.5 \pm 0.8$ | $34.5 \pm 0.8$ |
| $45.9 \pm 0.7$ | $41.6 \pm 0.7$ | $36.6 \pm 0.5$ |
| $47.4 \pm 0.4$ | $42.0 \pm 0.6$ | $34.5 \pm 0.8$ |
| $45.5 \pm 0.7$ | $\mathbf{37.4 \pm 0.5}$ | $31.8 \pm 0.6$ |
| $50.1 \pm 0.6$ | $45.9 \pm 0.9$ | $36.7 \pm 0.8$ |
| $45.0 \pm 0.7$ | $38.6 \pm 0.6$ | $\mathbf{30.3 \pm 0.5}$ |

(b) CIFAR-100

| M=0.2k | M=0.5k | M=1k |
|---|---|---|
| $72.8 \pm 1.7$ | $63.1 \pm 2.4$ | $55.8 \pm 2.6$ |
| $73.6 \pm 1.0$ | $63.4 \pm 1.6$ | $52.4 \pm 2.0$ |
| $71.7 \pm 2.1$ | $60.1 \pm 1.7$ | $53.5 \pm 2.5$ |
| $73.6 \pm 1.4$ | $58.3 \pm 2.2$ | $49.0 \pm 2.2$ |
| $71.1 \pm 1.8$ | $59.1 \pm 1.5$ | $50.4 \pm 1.5$ |
| $72.4 \pm 1.9$ | $58.8 \pm 1.4$ | $\mathbf{47.9 \pm 1.6}$ |

(c) CIFAR-10

Table C.2: Ablation analysis. Average Forgetting (lower is better). Memory buffer size M.

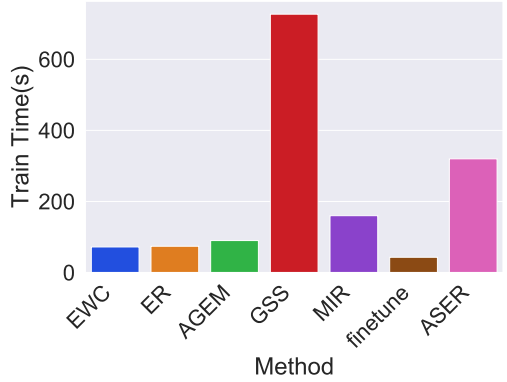



Figure B.1: Training time comparison for CIFAR-100

- **ASV$_\mu$-ret**: Use (6) scoring function for *MemoryRetrieval* while using reservoir sampling for *MemoryUpdate*.

Table C.1 compares the average accuracy of these variations. As for CIFAR-10, we can see that all SV-based methods improve upon ER. In particular, ASER and ASER$_\mu$ show the largest improvements, suggesting the effectiveness of the combination of the KNN-SV based *MemoryRetrieval* and *MemoryUpdate*. For the other two datasets, it turns out that SV-upd is a powerful *MemoryUpdate* method. Compared to GSS (Aljundi et al. 2019b) which suggests another *MemoryUpdate* method, we observe significant performance boosts (see Table 1). In these two datasets, ASV($_\mu$)-ret methods and ER perform comparably. However, we note that we can further fight the forgetting when *MemoryRetrieval* and *MemoryUpdate* are used together (Table C.2). In summary, ASER with KNN-SV *MemoryUpdate* performs competitively or better than the variations, underscoring the importance of SV-based methods for both *MemoryUpdate* and *MemoryRetrieval*.

## D Dataset Detail

Table D.1 shows the summary of the datasets used for the experiments. For a fair comparison, the classes in each task and the order of tasks are fixed in all experiments. For Split CIFAR-10, the first task contains class [0, 1], the second task contains class [2, 3], and so on. For Split CIFAR-100, similar to Split CIFAR-10, the first task contains class [0, 1, . . . , 9], the second task contains class [10, 11, . . . , 19] and so on.

In original Mini-ImageNet, 100 classes are divided into 64, 16, and 20 classes respectively for meta-training, meta-validation, and meta-test (Vinyals et al. 2016). For Split Mini-ImageNet, we firstly combine 64, 16, and 20 classes into one dataset. The first task contains the first 10 classes; the second task contains the next 10 classes, and so on.

## E Detail of Experiments

We use a reduced ResNet18, similar to (Chaudhry et al. 2019b; Lopez-Paz and Ranzato 2017), as the base model for all datasets, and the network is trained via cross-entropy loss with SGD optimizer. Note that several replay-based SOTA continual learning algorithms have also used the simple SGD (Aljundi et al. 2019a,b; Chaudhry et al. 2019a,b; Lopez-Paz and Ranzato 2017). For all experiments, we use the learning rate of 0.1 following the same setting as in Aljundi et al. (2019a). The mini-batch size is 10 and the size of the mini-batch retrieved from memory is also set to 10 irrespective of the size of the memory. Since we apply the online setting, the model only sees each batch once, so the number of epochs is set to 1 for all experiments.

We have used the memory size $M$ = 1k, 2k and 5k for Mini-ImageNet and CIFAR-100, while $M$ = 0.2k, 0.5k and

| | Split Mini-ImageNet | Split CIFAR-100 | Split CIFAR-10 |
|---|---|---|---|
| num. of tasks | 10 | 10 | 5 |
| image size | 3x84x84 | 3x32x32 | 3x32x32 |
| num. of classes per task | 10 | 10 | 2 |
| num. of training images per task | 4800 | 4500 | 9000 |
| num. of validation images per task | 600 | 500 | 1000 |
| num. of testing images per task | 600 | 1000 | 1000 |

Table D.1: Dataset statistics

1k for CIFAR-10. As for CIFAR-10, we use the same memory sizes as in MIR (Aljundi et al. 2019a) (0.4%, 1.1% and 2.2%); however, we found that they used disproportionately bigger memory for Mini-ImageNet (20% of the training data). One of the key desiderata of continual learning for deployment is limited memory footprint (Chaudhry et al. 2019a,b; Farquhar and Gal 2018; Parisi et al. 2019). Hence, we instead use smaller sizes of memory for both CIFAR-100 and Mini-ImageNet (around 2%, 4% and 10% of the training data) that better reflect real-world use cases with a high ratio of data to memory.

As for the hyperparameters of baselines, we tune the number of samples used for computing maximal gradients cosine similarity for GSS. For MIR, we tune the number of subsamples used to apply the MIR search criterion using the validation sets.

As for the hyperparameters of baselines, we use the validation sets to tune the number of samples (S) used for computing maximal gradients cosine similarity for GSS; the number of subsamples (C) used to apply the MIR search criterion for MIR. We have tuned two hyperparameters for ASER: the number of candidate samples ($N_c$) and the number of neighbors ($K$) for KNN-SV computation. Details of the datasets used in the experiment are shown in Table D.1 in Appendix D. We have summarized the hyperparameters used in the experiments in Table E.1.

The code to reproduce all results can be found in the attached zip file.

# F Discrepancy of CIFAR-10 Result for MIR between Original Paper and our Work

In the official repository of MIR, the authors apply a trick to improve performance that is not mentioned in the original paper. Specifically, during *MemoryRetrieval*, the trick excludes the memory samples from the current task. Note that to apply this trick, task identity is required during training. Our experimental results for MIR shown in Table 1 are based on the implementation of the original paper and therefore we have not applied this trick.

To understand the effect of this trick, we apply it to both MIR and our proposed ASER & ASER$_\mu$. As we can see in Table F.1, the trick indeed improves the results of MIR, especially in Mini-ImageNet (when M=5k) and CIFAR-10 (when M=1k). Nevertheless, this trick is not always useful. For example, when the memory buffer is small, this trick does not help and, in some cases, shows detrimental effects. In contrast, the trick does not have too much effect on our proposed ASER & ASER$_\mu$. Most results are very similar to the ones without the trick.

| Method | Hyperparameters | | |
|---|---|---|---|
| GSS | S=20 | S=10 | S=10 |
| MIR | C=100 | C=50 | C=50 |
| ASER | $K = 7$ $N_c = 200$ | $K = 1$ $N_c = 350$ | $K = 3$ $N_c = 90$ |
| ASER$_\mu$ | $K = 3$ $N_c = 250$ | $K = 3$ $N_c = 150$ | $K = 3$ $N_c = 90$ |
| | (a) Mini-ImageNet | (b) CIFAR-100 | (c) CIFAR-10 |

Table E.1: Hyperparameters selected for baselines and two variations of ASER.

| Method | M=1k | M=2k | M=5k | M=1k | M=2k | M=5k | M=0.2k | M=0.5k | M=1k |
|---|---|---|---|---|---|---|---|---|---|
| MIR | $8.1 \pm 0.3$ | $11.2 \pm 0.7$ | $15.9 \pm 1.6$ | $11.2 \pm 0.3$ | $14.1 \pm 0.2$ | $21.2 \pm 0.6$ | $28.3 \pm 1.6$ | $35.6 \pm 1.2$ | $42.4 \pm 1.5$ |
| MIR$^t$ | $7.7 \pm 0.6$ | $10.2 \pm 0.7$ | $17.8 \pm 1.0$ | $11.2 \pm 0.3$ | $14.5 \pm 0.3$ | $21.9 \pm 0.5$ | $28.0 \pm 1.1$ | $36.9 \pm 1.7$ | $44.9 \pm 0.9$ |
| ASER | $11.7 \pm 0.7$ | $\mathbf{14.4 \pm 0.4}$ | $18.2 \pm 0.7$ | $12.3 \pm 0.4$ | $14.7 \pm 0.7$ | $20.0 \pm 0.6$ | $27.8 \pm 1.0$ | $36.2 \pm 1.1$ | $43.1 \pm 1.2$ |
| ASER$_\mu$ | $\mathbf{12.2 \pm 0.8}$ | $14.8 \pm 1.1$ | $18.2 \pm 1.1$ | $\mathbf{14.0 \pm 0.4}$ | $17.2 \pm 0.5$ | $21.7 \pm 0.5$ | $26.4 \pm 1.5$ | $36.3 \pm 1.2$ | $43.5 \pm 1.4$ |
| ASER$^t$ | $11.3 \pm 0.4$ | $13.8 \pm 0.5$ | $18.0 \pm 1.0$ | $13.2 \pm 0.5$ | $16.1 \pm 0.3$ | $20.7 \pm 0.5$ | $27.6 \pm 1.4$ | $35.8 \pm 1.9$ | $42.4 \pm 1.4$ |
| ASER$_\mu^t$ | $12.2 \pm 0.5$ | $\mathbf{15.3 \pm 0.5}$ | $\mathbf{18.4 \pm 0.9}$ | $13.8 \pm 0.3$ | $\mathbf{17.3 \pm 0.5}$ | $21.5 \pm 0.7$ | $26.2 \pm 1.3$ | $36.9 \pm 1.3$ | $44.0 \pm 1.1$ |
| | (a) Mini-ImageNet | | | (b) CIFAR-100 | | | (c) CIFAR-10 | | |

Table F.1: The superscript $^t$ means using the "exclude current task samples" trick, namely excluding the samples from current task during *MemoryRetrieval*