GRADIENT IMBALANCE AND SOLUTION IN ONLINE CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing techniques for online continual learning (online CL) are based on *experience-replay*. In this approach, a *memory buffer* is used to save some data from past tasks for dealing with catastrophic forgetting. In training, a small batch of data from the data stream of the current task and some sampled data from the memory buffer are used to jointly update or train the model. In this paper, we study the experience-replay based approach from the angle of *gradient imbalance*. We first investigate and analyze this phenomenon experimentally from two perspectives: imbalance of samples introduced by experience-replay and sequence of classes introduced by incremental learning. To our knowledge, this problem has not been studied before and it significantly impacts the performance of online CL. Based on the observations from experiments and theoretical analysis, a new method, called GAD, consisting a new learning strategy and a novel loss function is proposed to deal with the problem. Empirical evaluation shows that GAD helps improve the online CL performance by more than 11% in accuracy.

1 INTRODUCTION

Continual learning (CL) is an important research topic as a truly intelligent agent should learn continually or incrementally to gain more and more knowledge. CL is defined as learning a sequence of tasks incrementally. It has two main settings, *batch CL* and *online CL*. In batch CL, when a task arrives, the full training data of it is available and the training process can go through the full data multiple times or epochs. In contrast, in online CL, the data of each task comes gradually in a data stream. The learning system can go through the data only once, i.e., training in one epoch incrementally with each small batch of arriving data from the stream. This paper focuses online CL with the goal of overcoming the well-known catastrophic forgetting (CF) problem (McClelland et al., 1995). There are also several CL scenarios. This paper works in the *class incremental learning* (CIL) scenario, which learns a sequence of tasks with non-overlapping classes. Each task learns a set of unique classes together. The resulting model is used to classify test samples from any class without providing any task related information.

Although a large number of diverse approaches have been proposed for batch CL, almost all existing online CL systems are based on *experience-replay*, or simply *replay*. In a replay method in online CL, a memory buffer is used to store a small number of previous training samples that the system has seen so far. Whenever a small batch of new data denoted by X^{new} is accumulated from the data stream, a small batch of samples is retrieved from the memory buffer, denoted by X^{buf}, and used to jointly train or update the current model while also dealing with CF. However, little work has been done to analyze the behaviors of the approach and its training process. In this work, we perform a gradient-based analysis, which reveals two important issues that hamper the performance of the replay approach. This first issue is gradient imbalance in terms of the number of samples per class (also called *data imbalance* or *class imbalance*). Since the batch sizes of the new data X^{new} from the data stream and the sampled data X^{buf} from the memory buffer are both fixed, if the system has already learned many tasks, the average number of samples in each previous class in X^{buf} will be extremely small, much smaller than that of each class in X^{new}. This causes gradient (data) imbalance, which is harmful to learning because the previous learned knowledge will not be well adjusted to adapt to the new task to re-establish good decision boundaries among old classes, and between the old classes in previous tasks and the new classes in the current/new task. We analyze this issue from the angle of gradients, which indicates a problem with the training process. The

second issue is the gradient imbalance caused by the incremental learning of tasks and their classes. This is an inherent problem of CL. That is, even if we can have balanced data for each class, it is still problematic because the training emphasizes the new samples (detailed in Sec. 4.2), which also causes gradient imbalance and poor performance. To our knowledge, this gradient based analysis has not done before. Although some researchers have worked on data imbalance (the first issue) in batch CL, the problem in online CL is quite different. We will compare them in detail in Sec. 2.

Based on the analysis, we propose a novel method, called GAD (*Gradient-Adjusted Dynamic online CL*), to deal with the two types of gradient imbalance. Specifically, we separate the batch of training data in each iteration into two groups: one for learning knowledge from the new task and the other for preserving and adjusting the previously learned knowledge. A novel loss function (called *GAD loss*) is also designed to deal with the second issue. The loss function can be dynamically adjusted as more and more tasks are learned to deal with the accumulated gradient imbalance and task specific gradient imbalance (detailed in Sec. 5) to achieve much better performances.

Our main contributions are as follows: **1)** To our knowledge, this is the first work that analyzes the behavior of the replay-based approach in online CL from the angle of *gradient imbalance*, which identifies two issues of gradient imbalance: one is introduced by the replay method itself and the other by incremental learning. **2)** It proposes a new method, called GAD, with a novel loss function to deal with the two imbalances, which is entirely different from the methods used in batch CL to address the data imbalance problem. Experimental results show that the proposed GAD outperforms the best baselines by 13.5% on CIFAR10, 11.5% on CIFAR100, and 11.5% on TinyImageNet.

2 Related Work

Online Continual Learning: Due to fast data streams in online CL, the model can see the data only once in training, i.e., training in one epoch. Existing online CL methods mainly use the replay approach. They typically use different methods to update the memory buffer and to retrieve samples from the memory when a new batch of data arrives from the data stream. For example, ER (Chaudhry et al., 2019a) simply randomly samples the replay data X^{buf} from the memory buffer. MIR (Aljundi et al., 2019a) uses a retrieval strategy that chooses replay samples whose loss increase the most after a current batch update. ASER (Shim et al., 2021) uses the Shapley value theory to design memory update/retrieval strategies. GDumb (Prabhu et al., 2020) uses a sampler to greedily store samples from the data stream, while using all the samples in the memory in training. Our system also uses a replay method, but our method is very different as we identify two issues (Sec. 1) with the replay approach and design a new training strategy and a novel loss function to deal with them.

Contrastive Learning: Recently contrastive learning has achieved significant improvements in many problems. The main idea is to align the representations of the anchor and its positive samples while pushing away the representations of negative samples. Chen et al. (2020) used data augmentation to create positive pairs and InfoNCE-type loss to calculate the contrastive loss in the unsupervised setting. Khosla et al. (2020) provided a supervised version of contrastive learning. Mai et al. (2021) proposed a recent online replay method (SCR) that uses supervised contrastive loss to learn feature representations. Our GAD system also exploits contrastive learning.

Class Imbalance: In a replay method, the data from previous tasks are from a limited-sized memory buffer but the data of the new task is often large. This causes class (or data) imbalance between the classes of the current task and those of the previous tasks. Several researchers have noticed the issue in batch CL. To solve this problem, Castro et al. (2018) added an additional fine-tuning stage with a small learning rate and a balanced subset of samples. Wu et al. (2019) added a bias correction layer after the last fully connected (FC) layer to correct the strong bias towards the classes in the new task. Zhao et al. (2020) provided a weight alignment method to correct the biased weights in the FC layer after the normal training process. Mittal et al. (2021) combined several components and applied a loss to balance intra-task and inter-task learning. However, these systems all need the full data of the current task to be available upfront to perform two learning stages and/or train multiple epochs to address the data imbalance problem, which are not suitable for online CL as online CL does not have all the data of a task available when the task first arrives in the data stream. To our best knowledge, we are the first to investigate this kind of class or data imbalance in the online CL setting. More importantly, we identify the second of imbalance, which has not been studied before. We have designed a new strategy and a novel loss function to deal with them.

3 PRELIMINARY

Problem description. This work is concerned with learning a sequence of tasks in the online class-incremental learning (CIL) setting. Let the sequence of tasks with their labeled training data be $T^{(1)}, T^{(2)}, T^{(3)}, ...,$, where $T^{(t)} = \{(x_k^{(t)}, y_k^{(t)})\}_{k=1}^{|T^{(t)}|}$ and $L^{(t)}$ be the set of classes of task t. We use $|T^{(t)}|$ to denote the number of training examples in task $T^{(t)}$. In CIL, the sets of classes of any two different tasks are disjoint. The training data of each task comes in a data stream and is only trained incrementally in one epoch, i.e., the data is seen only once by the learning algorithm.

Model architecture and training setting. The model consists of a feature extractor h_θ with the parameter set θ and a classifier f_ϕ with the parameter set ϕ. It uses a replay method with a memory buffer $\mathcal{M}$. Whenever a small batch of new data X^{new} from the stream is accumulated, it is trained jointly with a small batch of data X^{buf} sampled from the memory buffer $\mathcal{M}$ to update the model in one training iteration. The model produces the logits $o(x; \theta, \phi) = f_\phi(h_\theta(x))$, which are used to calculate the loss or to predict the class label of each input data.

4 THE GRADIENT IMBALANCE PROBLEM

In continual learning (CL), each task is usually learned by minimizing the cross-entropy loss, $\mathcal{L}_{ce}$:

$$\mathcal{L}_{ce}(o(x; \theta, \phi)) = -\sum_{j=1}^{|C_{seen}|} l_{c_j} \log(p^{c_j}), \qquad p^{c_j} = \frac{e^{o_{c_j}}}{\sum_{s=1}^{|C_{seen}|} e^{o_{c_s}}} \tag{1}$$

where $|C_{seen}|$ is the number of classes that the model has seen, $l_{c_j} \in \{0, 1\}$ is the one-hot ground-truth label of class c_j, and $o(x; \theta, \phi) = [o_{c_1}, o_{c_2}, ..., o_{c_{|C_{seen}|}}]$ is the set of logit values of input x. Given a training sample x^i of class c_i, the gradients on logits and the classifier f_ϕ are given by

$$\begin{aligned}
&\frac{\partial \mathcal{L}_{ce}(o(x^i; \theta, \phi))}{\partial o_{c_i}} = p^{c_i} - 1, \qquad \frac{\partial \mathcal{L}_{ce}(o(x^i; \theta, \phi))}{\partial o_{c_j}} = p^{c_j} \\
&\frac{\partial \mathcal{L}_{ce}(o(x^i; \theta, \phi))}{\partial \phi_{c_i}} = \frac{\partial \mathcal{L}_{ce}(o(x^i; \theta, \phi))}{\partial o_{c_i}} \frac{\partial o_{c_i}}{\partial \phi_{c_i}} = (p^{c_i} - 1) f_\theta(x^i) \\
&\frac{\partial \mathcal{L}_{ce}(o(x^i; \theta, \phi))}{\partial \phi_{c_j}} = \frac{\partial \mathcal{L}_{ce}(o(x^i; \theta, \phi))}{\partial o_{c_j}} \frac{\partial o_{c_j}}{\partial \phi_{c_i}} = p^{c_j} f_\theta(x^i)
\end{aligned} \tag{2}$$

where $\phi = [\phi_{c_1}, \phi_{c_2}, ..., \phi_{c_{|C_{seen}|}}] \in R^{d \times |C_{seen}|}$, and d is the dimension of weights. We can view f_ϕ as the union of the classifiers of all classes. From Eq. 2, we observe that x^i gives its true logit o_{c_i} a positive gradient and the other logits o_{c_j} negative gradients. The negative gradients help the model to adjust wrong classifications, and the positive gradient encourages the model to output bigger probability on the true class. Both are important for the model to learn effectively. However, in CL, the model cannot revisit the training data of previous tasks when a new task arrives. So all the gradients for the classifiers of previous classes are negative gradients from the new data. That causes the catastrophic forgetting (CF). From the perspective of logits, the model tends to output smaller probabilities on the previous classes. Then if we test the model with samples from some previous classes, the model tends to view them as samples from the new classes, and to view the knowledge belonging to previous classes as background or noise. Thus the learned knowledge for previous classes is forgotten. That is an intrinsic problem of continual learning. Below, we experimentally study it from the perspective of negative and positive gradients. We introduce the setup first.

Tasks and methods: We conduct experiments using two dataset settings: (1) Split CIFAR100, where we divide the CIFAR100 dataset into 10 tasks with 10 unique classes per task, and (2) Split TinyImagenet, where we divide the TinyImageNet dataset into 20 tasks with 10 different classes per task. As we are interested in online CL, we run each task one epoch.

Model, optimizer and batch size: We use the full size ResNet-18 to perform our experiments. To provide a more general analysis, we run with both the SGD and Adam optimizers. For learning rate, we follow (Aljundi et al., 2019a) and set it as 0.1 for the SGD optimizer. To ensure good performance, we search and set the learning rate as 0.001 for the Adam optimizer. For batch size, we follow (Aljundi et al., 2019a) and set it to 10 for X^{new} and 10 as well for the buffer batch X^{buf},

which is randomly sampled from the memory buffer. Since ER (Experience Replay) is a basic replay method for online CL, we conduct our experiments using it.

Metrics: When the model is learning the new task $T^{(n)}$, for a class $c_i \in L^{(n)}$, we define $P(T^{(n)}, c_i)$ (or $N(T^{(n)}, c_i)$) as the *total positive* (or *negative*) *gradient* that c_i's logit receives for this task:

$$P(T^{(n)}, c_i) = \sum_{k=1}^{|T^{(n)}|} (p_k^{c_i} - 1) \cdot \mathbb{I}(y_k^{(n)} = c_i), \qquad N(T^{(n)}, c_i) = \sum_{k=1}^{|T^{(n)}|} p_k^{c_i} \cdot \mathbb{I}(y_k^{(n)} \neq c_i) \tag{3}$$

where $\mathbb{I}$ is the indicator function. Similarly, we define $P(T^{(n)}, L^{(t)}) = \sum_{i=1}^{|C_{seen}|} P(T^{(n)}, c_i) \cdot \mathbb{I}(c_i \in L^{(t)})$ (or $N(T^{(n)}, L^{(t)}) = \sum_{i=1}^{|C_{seen}|} N(T^{(n)}, c_i) \cdot \mathbb{I}(c_i \in L^{(t)})$) as the *total positive* (or *negative*) *gradient* that the logits of the class set $L^{(t)}$ of a previous task t receive in training a new task $T^{(n)}$. We use these metrics and their variants to analyse the behavior of the online CL model described above.

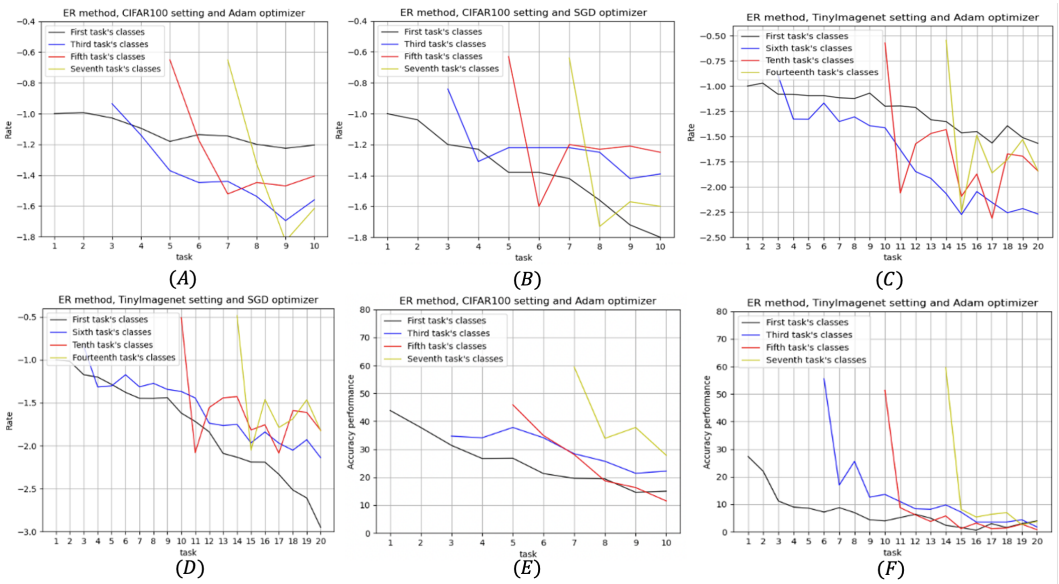

Figure 1: The *NP* rate (*rate* in the figures) of the CIFAR100/TinyImagnet experiments with the two optimizers. The buffer size is 1000. We choose four different tasks and plot their *NP* rates as subsequent tasks are learned. In the last two figures, we separately report the four tasks' average test accuracy after training a new task in the CIFAR100/TinyImagnet experiment.

4.1 Gradient imbalance caused by Class data imbalance

We first study how the number of training samples in each class influences the gradients of the classes. The replay method mitigates CF by replaying some previous data in the memory buffer. The batch size N^{buf} for the buffer batch X^{buf} and the batch size N^{new} for the new data batch X^{new} are usually fixed. As the number of previous classes grows with more tasks learned, the number of sampled data for each previous class $\frac{N^{buf}}{\bigcup_{t=1}^{n-1} |L^{(t)}|}$ in X^{buf} gets smaller, but the number of samples for each new class $\frac{N^{new}}{|L^{(n)}|}$ in X^{new} of the new task remains unchanged. Then we have $\frac{N^{new}}{|L^{(n)}|} > \frac{N^{buf}}{\bigcup_{t=1}^{n-1} |L^{(t)}|}$ and the samples from the previous classes and the new data classes can become highly imbalanced. The negative gradients of the classifiers for the previous classes can surpass their positive gradients. We verify this by calculating the *Negative-Positive*) (NP) rate: $NP = \frac{N(T^{(n)}, L^{(t)})}{P(T^{(n)}, L^{(t)})}$, where $t \leq n$. We show the NP results of the classes of a few tasks in Figure 1 as more subsequent tasks are learned.

We observe from Figures 1 (A), (B), (C) and (D) that when the model is trained with the new task, the *NP* rates for the classes of the previous tasks get smaller than -1 and tend to decrease. This shows that the negative gradient surpasses the positive gradient of the previous classes. Another interesting observation is that the *NP* rate of the classes of the first task is usually near -1. This is

because the randomly initialized model using the training data of the first task to learn features of all levels or layers. The lower-level features are usually less forgotten (Ramasesh et al., 2020). Thus the model has a bias toward the classes of the first task. We also observe from (E) and (F) that the tendency of the test accuracy performance of the previous classes is similar to the tendency of their *NP* rates. We will propose a method in the next section to solve the data imbalance problem.

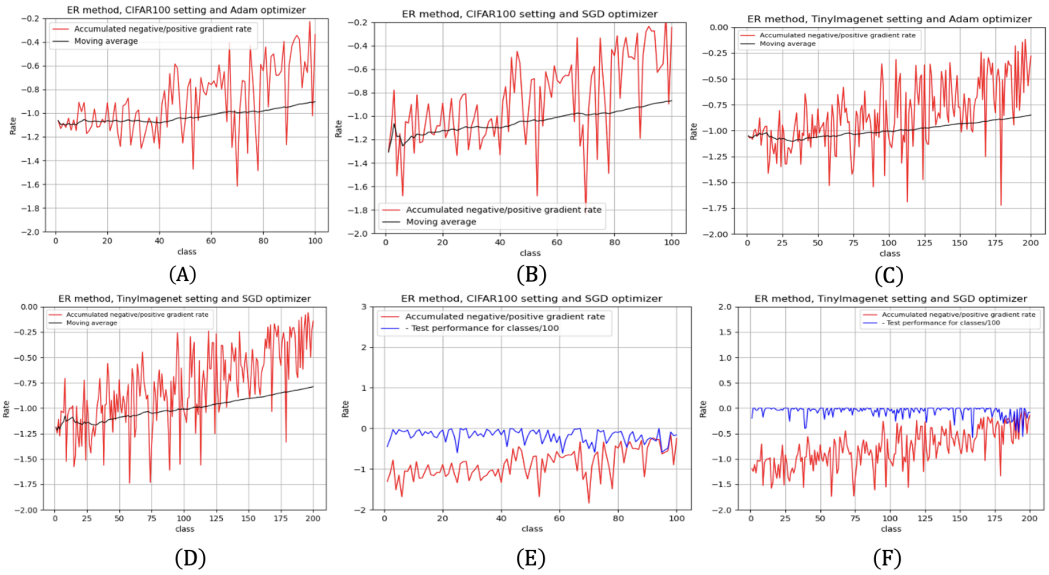

Figure 2: The accumulated A-NP rate (*rate* in the figures) for each class (x-axis) after the last task is trained. The buffer size is 1000. In the last two sub-figures, we report the test accuracy of each class and the accumulated gradient (A-NP) rate after the training of the last task.

4.2 Gradient imbalance caused by sequential learning of classes

This subsection shows that gradient imbalance also occurs when the number of samples used in each class is the same in each training iteration. To see this, we use the *accumulated negative and positive gradient rate* (A-NP rate) for an old class $c_i \in L^{(j)}$ and the current task $T^{(n)}$ ($j \leq n$)

$$\textit{A-NP}(T^{(n)}, c_i) = \frac{\sum_{t=j}^{n} N(T^{(j)}, c_i)}{\sum_{t=j}^{n} P(T^{(j)}, c_i)} \tag{4}$$

A-NP$(T^{(n)}, c_i)$ is used to show the gradient imbalance of each class. We conduct a new experiment that uses the same number of training samples for each class in each training iteration. Specifically, we fix $N^{new} + N^{buf} = 20$. We use the rate between $|L^{(n)}|$ and $\sum_{t=1}^{n} |L^{(t)}|$ to decide the number of samples in X^{new} in training (the rest are discarded), i.e., $N^{new} = max(int(20 \cdot \frac{|L^{(n)}|}{\sum_{t=1}^{n} |L^{(t)}|}), 1)$ where $int(\cdot)$ returns the nearest integer of a given number and $N^{buf} = 20 - N^{new}$. We use N^{buf} to sample previous tasks data in the memory buffer. In this way, the number of training data for each class is equal.

We plot the A-NP rate of each class in Figure 2(A), (B), (C) and (D) after the last task is learned. We observe that A-NP rates of newer classes are larger than those of older classes. The A-NP rates of the new classes are also greater than -1. The reasons are: on the one hand, the data of the new task have more hard samples to train since they are new and the data in the memory buffer may have been trained many times before. On the other hand, since the model has trained the data of previous classes, the loss of the new task data is comparatively larger than the loss of previous task data, which encourages the model to classify data into the new classes to reduce loss. Thus the positive gradients of old classes decrease and the positive gradients of new classes increase. The fact that the accumulated negative gradients are greater than the accumulated positive gradients (i.e., *A-NP* < -1) in old classes makes the model belittle old classes. That is a natural problem of continual learning and it is caused by the incremental learning process.

The second observation is that the moving average of A-NP rates tend to monotonically increase except for the classes of the first task. The reason for the classes of the first task to have comparatively higher A-NP rates is the same as we mentioned in Sec. 4.1 (the model has a bias towards the classes of the first task). The figures show that there is a gradient imbalance across classes. The earlier classes (except those in the first task) tend to suffer more from imbalance than later tasks. We observe from (E) and (F) that the tendency of the negated (times -1) test accuracy performance of each class is similar to the tendency of the *A-NP* rate.

5 THE PROPOSED METHOD

Based on the above observations, we now propose our method GAD to mitigate the gradient imbalance problems. It consists of two parts: (1) a sampling strategy that produces two groups of data for each training iteration, one for learning new classes and one for consolidating previous knowledge to deal with gradient imbalance caused by class (or data) imbalance (Sec. 4.1), and (2) a new loss function to mitigate the gradient imbalance caused by sequential learning of classes (Sec. 4.2).

Group 1: This group aims to maintain the boundaries of previous classes and to establish the new boundaries between new classes and previous classes. To resolve the imbalance of samples introduced by replay, we use the sampling strategy proposed in Sec. 4.2. Then we obtain a mix set of new data and old data sampled from the memory buffer, called X^{mix}. X^{mix} can be regarded as samples from the uniform joint distribution for all classes, including both new classes and previous classes.

Group 2: This group aims to establish the new decision boundaries between the classes in the current/new task. Group 2 is X^{new}. We assume each class has the same number of samples in X^{new}.[1] So there is no class imbalance caused by replay.

Loss for group 1: we propose the GAD loss $\mathcal{L}_{GAD}$ for data in group 1. When the model learns the current task $T^{(n)}$, for the mixed batch $X^{mix} = \{x_k, y_k\}_{k=1}^{N^{mix}}$, its batch loss $\mathcal{L}^b_{GAD}$ is

$$\mathcal{L}^b_{GAD}(X^{mix}) = -\frac{\sum_{k=1}^{N^{mix}} w_{y_k} \mathcal{L}_{GAD}(o(x_k;\theta,\phi))}{N^{mix}}, \qquad \mathcal{L}_{GAD}(o(x_k;\theta,\phi)) = \sum_{j=1}^{|C_{seen}|} l_{c_j} \log(\hat{p}^{c_j})$$
$$\hat{p}^{c_j} = \frac{v_{y_k c_j} e^{o_{c_j}}}{\sum_{s=1}^{|C_{seen}|} v_{y_k c_s} e^{o_{c_s}}} \tag{5}$$

where

$$w_{y_k} = \begin{cases} 1 & y_k \in L^{(n)} \\ |\textit{A-NP}(T^{(n)}, y_k)|^* & y_k \notin L^{(n)} \end{cases}, \qquad v_{y_k c_j} = \begin{cases} 1 & c_j \neq y_k \vee y_k \in L^{(n)} \\ \frac{1}{|\textit{NP}(T^{(n)}, y_k)|^*} & c_j = y_k \wedge y_k \notin L^{(n)} \end{cases} \tag{6}$$

where $|\cdot|^*$ gives the absolute value. When the model is training task $T^{(n)}$ and the task has not ended, we use the gradients of all the data of the task seen so far to calculate *A-NP*$(T^{(n)}, y_k)$ and *NP*$(T^{(n)}, y_k)$. Compared to the original cross entropy loss, $\mathcal{L}_{GAD}$ has two advantages: (1) When the new task arrives, we do not need to assume that we have the entire training set for calculating some statistics or an exemplar set. Our method is suitable for online CL. (2) The model sets the value of w_{y_k} and $v_{y_k c_j}$ according to *A-NP* rates and NP rates automatically as they vary dynamically based on the current gradient imbalance situation as more tasks are learned. No need to set any hyper-parameters or threshold manually. We justify the new loss by the following gradient analysis.

Assuming x^i is a data sample of class c_i, the gradient on each logit of $o(x^i;\theta,\phi)$ is given by:

$$\frac{\partial \mathcal{L}_{ce}(o(x^i;\theta,\phi))}{\partial o_{c_i}} = \begin{cases} p^{c_i} - 1 & c_i \in L^{(n)} \\ |\textit{A-NP}(T^{(n)}, c_i)|^* \cdot \left(\frac{\frac{e^{o_{c_i}}}{|\textit{NP}(T^{(n)}, c_i)|^*}}{\sum_{s=1 \bigcap s\neq i}^{|C_{seen}|} e^{o_{c_s}} + \frac{e^{o_{c_i}}}{|\textit{NP}(T^{(n)}, c_i)|^*}} - 1\right) & c_i \notin L^{(n)} \end{cases} \tag{7}$$

$$\frac{\partial \mathcal{L}_{ce}(o(x^i;\theta,\phi))}{\partial o_{c_j}} = \begin{cases} p^{c_j} & j \neq i \bigcap c_i \in L^{(n)} \\ |\textit{A-NP}(T^{(n)}, c_i)|^* \cdot \frac{e^{o_{c_j}}}{\sum_{s=1 \bigcap s\neq i}^{|C_{seen}|} e^{o_{c_s}} + \frac{e^{o_{c_i}}}{|\textit{NP}(T^{(n)}, c_i)|^*}} & j \neq i \bigcap c_i \notin L^{(n)} \end{cases} \tag{8}$$

[1] If the original new training set is not class balanced. We can use over-sampling/sub-sampling of the new class data from the memory buffer to deal with the problem.

When c_i is a class in the new task, its positive/negative gradients are not changed. When class c_i belongs to a previous task, if its accumulated negative gradient surpasses the accumulated positive gradients ($|\textit{A-NP}(T^{(n)}, c_i)|^* > 1$), based on Eq. 5 and Eq. 7, its sample weight w_i in the loss will be larger than 1 and the absolute value of its positive gradient will be increased and $|\textit{A-NP}(T^{(n)}, c_i)|^*$ will be decreased until $|\textit{A-NP}(T^{(n)}, c_i)|^* = 1$, or vice versa. The model can automatically adjust the loss weight w_i to balance the accumulated negative gradient and the accumulated positive gradient for each class. However, the accumulated gradient rate *A-NP* is calculated from all seen training data and so is not sensitive to short-term tendency. To capture short-term variations within a task, we introduce $\frac{1}{|\textit{NP}(T^{(n)},c_i)|^*}$ to the logit of c_i. From Eq. 7 and Eq. 8, we know that if the negative gradient of c_i within a task surpasses the positive gradient of c_i within a task ($\frac{1}{|\textit{NP}(T^{(n)},c_i)|^*} < 1$), the positive gradient will be adjusted by increasing $\frac{1}{|\textit{NP}(T^{(n)},c_i)|^*}$ until $\frac{1}{|\textit{NP}(T^{(n)},c_i)|^*} = 1$ and the negative gradient for the logits of other classes (not c_i) will be increased to help the model distinguish samples of other classes, and vice versa. We also use the supervised contrastive loss $\mathcal{L}_{sup}$ to help the representation learning. The total loss for group 1 is

$$\mathcal{L}_{g1} = \mathcal{L}_{GAD}(X^{mix}) + \mathcal{L}_{sup}(X^{mix}) \tag{9}$$

Loss for group 2: For group 2, we use cross entropy loss and $\mathcal{L}_{sup}$. To isolate the influence of the new classes on previous classes, for a data sample x in X^{new}, cross entropy is calculated as

$$\mathcal{L}_{\textit{separated-ce}}(o(x;\theta,\phi)) = -\sum_{j=1}^{|L^{(n)}|} l_{c_j} \log(p^{c_j}), \qquad p^{c_i=j} = \frac{e^{o_{c_j}}}{\sum_{s=1}^{|L^{(n)}|} e^{o_{c_s}}} \tag{10}$$

In this way, the negative gradients for previous classes are set to zero. As new classes in $T^{(n)}$ appear together, so there is no imbalance problem caused by the learning order of classes in a task. The sample number for each class is also the same (also see footnote 1).

Our method GAD jointly trains the two groups of data and updates their losses. The total loss is

$$\mathcal{L}(X^{mix} \bigcup X^{new}) = \mathcal{L}_{GAD}(X^{mix}) + \mathcal{L}_{sup}(X^{mix}) + \mathcal{L}_{\textit{separated-ce}}(X^{new}) + \mathcal{L}_{sup}(X^{new}) \tag{11}$$

In each group, the sample number of the classes is the same and the gradient imbalance introduced by the sequence learning is also mitigated. Experiments in the following section show that our method outperforms baselines significantly.

6 EXPERIMENTS

Evaluation data. Four image classification datasets are used in our experiments. 1) **MNIST** (LeCun et al., 1998) has 10 classes with 60,000 examples for training and 10,000 examples for testing. It is split into 5 disjoint tasks with 2 classes in each task. 2) **CIFAR10** (Krizhevsky & Hinton, 2009) has 10 classes with 50,000 for training and 10,000 for testing. It is split into 5 disjoint tasks with 2 classes per task. 3) **CIFAR100** (Krizhevsky & Hinton, 2009) has 100 classes with 50,000 for training and 10,000 for testing. It is split into 10 disjoint tasks with 10 classes per task. 4) **TinyImageNet** (Le & Yang, 2015) has 200 classes. It is split into 100 disjoint tasks with 2 classes per task. Each class has 500 training examples and 50 test examples. We use 100 tasks here to stress test the system.

Compared Baselines. GAD [2] is compared with 8 recent baselines (the first 7 of them are online CL systems and the last one is a batch CL system that deals with class imbalance. **AGEM** (Chaudhry et al., 2018) uses the average gradient information of the samples in the memory buffer to constrain the parameter updates. **ER** (Chaudhry et al., 2019a) is a replay method with random sampling in memory retrieval, and reservoir sampling in memory update. **MIR** (Aljundi et al., 2019a) is also a replay method. Given the estimated parameters update based on the new task, it retrieves replay samples that suffer from an increase in loss. **GSS** (Aljundi et al., 2019b) is a replay method that diversifies the gradients of the samples in the memory buffer. **ASER** (Shim et al., 2020) is based on the Shapley value theory. It first calculates the score for each memory buffer sample according to its ability to maintain stability and plasticity of the model. **GDumb** (Prabhu et al., 2020) uses a greedy and class balance strategy to update the memory buffer. It trains using only samples from the

[2]The code has been submitted in the supplementary file.

memory buffer. **SCR** (Mai et al., 2021) is a replay method that uses the supervised contrastive loss to train the model. To avoid bias toward new classes, it uses a Nearest Class Mean (NCM) classifier to predict the label for a new sample. All these 7 methods are online CL systems. **CCIL** (Mittal et al., 2021) is a latest replay method for *batch CL* that trains the model on a data-balance exemplar set constructed by randomly selecting an equal number of samples for each class (including the classes of the new task) before the learning of the new task. However, this method (like other batch CL methods (Zhao et al., 2020; Wu et al., 2019)) needs the full data for the new task at the beginning, which is not suitable for the stream setting of online CL.

6.1 Architecture, Augmentation, Training Details and Evaluation Protocol

Architecture. For MNIST, GAD employs a fully-connected network with two hidden layers as the feature extractor h_θ, each comprising of 400 ReLU units. We use a linear layer of size [400, 10] as the classifier f_ϕ and a linear layer of size [400, 128] as the projection head σ for contrastive learning. For CIFAR10, CIFAR100, and TinyImageNet, we follow (Buzzega et al., 2020) and use the full ResNet18 (not pre-trained) as the feature extractor h_θ with around 11 million trainable parameters.. Denoting C_{num} as the number of all classes, we employ a linear layer (size [dim_h, C_{num}]) as the classifier f_ϕ and a linear layer of size [dim_h, 128] as the feature projection head σ. For an input x, we use $f_\phi(h_\theta(x))$ to compute its cross entropy loss and $\sigma(h_\theta(x))$ to compute the supervised contrastive loss. For baselines, we use the same full ResNet18 backbone for fair comparisons.

Data augmentation and contrastive loss. For a fair comparison, the same data augmentation and supervised contrastive loss following (Tack et al., 2020) have been applied to all baselines and our GAD system. Data augmentation uses *horizontal-flip*, *random-resized-crop*, *random-gray-scale*, and *rotation* (Tack et al., 2020). This supervised contrastive loss $\mathcal{L}_{sup}$ improves ER, MIR, GDumb, ASER, and CCIL significantly by 6 to 21% and improves AGEM, GSS slightly by 2 to 4%.

Training and hyperparameter settings.

Like ER and many other online CL systems, GAD uses reservoir sampling for memory update. For all datasets, all baselines and GAD are trained with the Adam optimizer. We set the learning rate as 0.001 and fix the weight decay as 0.0001[3]. Following (Shim et al., 2020), we set each data batch (X^{new}) size N^{new} to 10 for all systems. We use X^{new} as the input of group 2. For the input of group 1 (X^{mix}), we set it to 64 in GAD. Specifically, we first sample $max(int(64 \cdot \frac{|L^{(n)}|}{\sum_{t=1}^{n}|L^{(t)}|}), 1)$ data points from X^{new} and sample $N^{buf} = 64 - N^{new}$ data points from the memory buffer, and then we concatenate them as the X^{mix}. Again for fair comparisons, for baselines, we set their memory buffer batch (X^{buf}) size as 64 (it does not change with tasks). We use the official codes[4] of the baselines and employ their default hyper-parameters. In fact, our experiments basically follow their settings. We also follow SCR (Mai et al., 2021) and set the temperature of supervised contrastive loss (Khosla et al., 2020) as 0.07. *We run all methods with one epoch for each task.*

Evaluation protocol. Accuracy is the evaluation metric. We first learn from the data stream of all tasks for each dataset, and then test the final model using the test data of all tasks. We report the average accuracy of all tasks from 15 random runs for each dataset. See Appendix for training time.

6.2 Results Analysis

Table 1 shows the accuracy results of our GAD system and all 8 baselines with various memory sizes on the four datasets. For all datasets and buffer sizes, GAD consistently outperforms all baselines by very large margins. For example, with the largest memory size for each dataset in Table 1, GAD outperforms the best baselines by 13.5% on CIFAR10, 11.5% on CIFAR100, 11.5% on TinyImageNet

[3] We use the Apex (A PyTorch Extension) `https://nvidia.github.io/apex/` to accelerate training for all methods. "opt_level" is "O1".

[4] The code of ER and MIR: `https://github.com/optimass/Maximally_Interfered_Retrieval`.
The code of ASER and SCR: `https://github.com/RaptorMai/online-continual-learning`.
The code of GDumb: `https://github.com/drimpossible/GDumb`.
The code of DER++: `https://github.com/aimagelab/mammoth`.
The code for AGEM: `https://github.com/facebookresearch/agem`.
The code for GSS: `https://github.com/rahafaljundi/Gradient-based-Sample-Selection`.
The code for CCIL:`https://github.com/sud0301/essentials_for_CIL`.

Table 1: Accuracy on MNIST (5 tasks), CIFAR10 (5 tasks), CIFAR100 (10 tasks) and TinyImageNet (100 tasks) with different memory buffer sizes *M*. All values are averages of 15 runs.

Method	MNIST			CIFAR10			CIFAR100			TinyImageNet		
M	*M*=0.1*k*	*M*=0.5*k*	*M*=1*k*	*M*=0.2*k*	*M*=0.5*k*	*M*=1*k*	*M*=1*k*	*M*=2*k*	*M*=5*k*	*M*=2*k*	*M*=4*k*	*M*=10*k*
AGEM	56.9±5.2	57.7±8.8	61.6±3.2	22.7±1.8	22.7±1.9	22.6±0.7	5.8±0.2	6.5±0.3	5.8±0.2	0.9±0.1	2.1±0.1	3.9±0.2
ER	78.7±0.4	88.0±0.2	90.3±0.1	49.7±0.6	55.2±0.6	59.3±0.2	15.7±0.3	22.0±0.5	14.4±0.9	4.7±0.5	10.1±0.7	11.7±0.2
MIR	87.0±0.1	92.0±0.1	94.0±0.1	51.0±0.4	57.0±0.4	62.1±0.3	20.1±0.2	25.1±0.4	33.1±0.2	8.0±0.3	14.0±0.3	15.5±0.4
GSS	70.4±1.5	80.7±5.8	87.5±5.9	26.9±1.2	30.7±1.3	40.1±1.4	11.1±0.2	13.3±0.5	17.4±0.1	3.3±0.5	10.0±0.2	10.5±0.2
ASER	62.6±0.9	79.0±0.2	86.5±0.5	32.3±0.5	44.3±0.7	50.1±0.4	16.4±0.3	22.2±0.5	30.1±0.3	6.0±0.2	14.2±0.2	20.1±0.2
GDumb	81.2±0.5	91.0±0.2	94.5±0.1	35.9±1.1	50.7±0.7	63.5±0.5	17.1±0.4	25.1±0.2	38.6±0.5	12.6±0.1	12.7±0.3	15.7±0.2
SCR	86.2±0.5	92.8±0.3	94.6±0.1	47.2±1.7	58.2±0.5	64.1±1.2	26.5±0.2	31.6±0.5	36.5±0.2	10.6±1.1	17.2±0.1	20.4±1.1
CCIL	86.4±0.1	92.8±0.2	94.0±0.1	50.5±0.2	55.3±0.54	59.8±0.3	18.5±0.3	19.1±0.4	20.5±0.3	5.6±0.9	7.0±0.5	15.2±0.5
GAD	90.4±0.1	91.5±0.1	96.2±0.1	62.3±0.4	71.5±0.3	77.6±0.5	32.7±0.2	40.5±0.2	50.1±0.5	17.7±0.4	24.3±0.2	31.9±0.4

Table 2: Ablation accuracy - average of 5 runs. *M* is the memory buffer size.

Dataset	no $\mathcal{L}_{GAD}$	no $\mathcal{L}_{separated\text{-}ce}$	union for $\mathcal{L}_{sup}$	no $\mathcal{L}_{sup}$	no group 2	no balanced sampling	no grouping
MNIST (*M*=1*k*)	95.2±0.1	92.1±0.2	95.2±0.1	94.2±0.1	95.7±0.1	95.9±0.1	95.3±0.1
CIFAR10 (*M*=1*k*)	75.0±0.3	65.5±0.6	75.6±0.4	68.7±1.2	75.2±0.4	75.3±0.5	74.3±0.2

and 1.7% on MNIST. The results also show that increased memory size results in increased accuracy. Due to space limitations, we give the results of forgetting rate and running time in Appendix.

6.3 Ablation Experiments

We conduct ablation experiments to analyze the contribution of various components and choices made in GAD using two datasets, MNIST and CIFAR10, with 1*k* (*M*=1*k*) memory. The results are given in Table 2.

(1). *Ablation study of training loss in GAD.* In experiment "no $\mathcal{L}_{GAD}$," we replace GAD loss with cross-entropy loss in group 1. In experiment "no $\mathcal{L}_{separated\text{-}ce}$", we replace the $\mathcal{L}_{separated\text{-}ce}$ loss with cross-entropy loss that considers logits of all classes. In experiment "union for $\mathcal{L}_{sup}$", we replace $\mathcal{L}_{sup}(X^{new})+\mathcal{L}_{sup}(X^{mix})$ with $\mathcal{L}_{sup}(X^{new} \bigcup X^{mix})$. In experiment "no $\mathcal{L}_{sup}$", we do not calculate the supervised contrastive loss for group 1 and group 2. From Table 2, we see that their performances are all poorer than GAD (Table 1). This is because in the first two cases, there is gradient imbalance caused by the incremental learning of classes. The learning of new classes dominates the learning process. So the model tends to predict samples from older classes to newer classes to reduce the loss. The poorer performance in the third experiment is because the data imbalance between the new classes and previous classes. The poor performance in the fourth experiment (no $\mathcal{L}_{sup}$, no contrastive loss) shows that the supervised contrastive loss improves the performance of our method and baselines greatly (see also "Data augmentation and contrastive loss" in Sec. 6.1).

(2). *Ablation study of data groups and the sampling strategy.* In experiment "no group2," we consider only the loss of group 1. In experiment "no balanced sampling," we replace the sampling strategy for X^{mix} with randomly sampling from the memory buffer. In experiment "no grouping," we first combine X^{new} (its size is 10) and X^{buf} (its size is 64) as one group, and then calculate the GAD loss and supervised contrastive loss. Table 2 shows that all these incomplete GAD systems are poorer than the full GAD. "no group2" does not do well as it discards the information in X^{new}. The poorer performances of "no balanced-sampling" and "no grouping" are due to data imbalance.

7 Conclusion

This paper investigated the most popular approach to online continual learning (CL), *experience replay* or *replay*, from a new perspective of *gradient imbalance*. It first analyzed this phenomenon experimentally from two perspectives: imbalance of data samples introduced by experience replay and sequence of classes introduced by incremental learning. To our knowledge, this problem has not been studied before in online CL and it significantly limits the online CL performance. Based on observations from the experiments and theoretical analysis, a new learning strategy and a new loss function have been proposed to deal with the problem. Empirical evaluation demonstrated that the new approach GAD helped improve the online CL performance substantially.

REFERENCES

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019b.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, pp. 233–248, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report TR-2009, University of Toronto, Toronto.*, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7:7, 2015.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3589–3599, June 2021.

James L McClelland, Bruce L McNaughton, and Randall C O'reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3513–3522, 2021.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *EECV*, pp. 524–540, 2020.

Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. *arXiv preprint arXiv:2007.07400*, 2020.

Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. *arXiv preprint arXiv:2009.00093*, 2020.

Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9630–9638, 2021.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Proceedings of 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, 2019.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13208–13217, 2020.

Table 3: Average forgetting rate. All numbers are the averages of 15 runs.

Method	MNIST			CIFAR10			CIFAR100			TinyImageNet		
M	*M*=0.1*k*	*M*=0.5*k*	*M*=1*k*	*M*=0.2*k*	*M*=0.5*k*	*M*=1*k*	*M*=1*k*	*M*=2*k*	*M*=5*k*	*M*=2*k*	*M*=4*k*	*M*=10*k*
AGEM	32.5±5.9	30.1±4.2	32.0±2.9	36.1±3.8	43.2±4.3	48.1±3.4	78.6±2.1	77.5±1.3	78.3±1.2	73.9±0.2	78.9±0.2	74.1±0.3
ER	22.7±0.5	9.7±0.4	6.7±0.5	42.0±0.3	26.7±0.7	20.7±0.7	65.1±1.3	59.3±0.9	60.0±1.6	68.2±2.8	66.2±0.8	67.2±0.2
MIR	16.0±0.2	8.0±0.2	5.0±0.2	34.0±0.3	19.2±0.3	18.0±0.3	64.0±0.2	54.0±0.4	50.0±0.4	62.1±0.6	60.1±0.1	59.5±0.2
GSS	26.1±2.2	17.8±5.22	10.5±6.7	75.5±1.5	65.9±1.6	54.9±2.0	73.4±4.2	69.3±3.1	70.9±2.9	72.8±1.2	72.6±0.4	71.5±0.2
ASER	33.6±1.1	18.2±1.1	11.7±1.5	67.0±1.8	44.3±0.8	41.5±0.6	65.0±0.2	52.2±0.9	53.2±0.1	63.7±0.7	56.0±0.3	46.4±0.1
GDumb	10.3±0.1	6.2±0.1	4.8±0.2	26.5±0.5	24.5±0.2	18.9±0.4	16.7±0.5	17.6±0.2	16.8±0.4	15.9±0.5	14.6±0.3	11.7±0.2
SCR	10.7±0.1	4.7±0.1	4.0±0.2	41.3±0.1	31.5±0.2	24.7±0.4	17.5±0.2	11.6±0.5	5.6±0.4	19.4±0.3	15.4±0.3	14.9±0.7
CCIL	14.1±0.1	7.7±0.1	4.8±0.1	18.6±0.1	16.5±0.4	12.5±0.8	16.7±0.5	16.1±0.3	17.5±0.2	59.4±0.3	56.2±1.3	48.9±0.6
GAD	8.1±0.1	2.5±0.1	1.4±0.1	29.1±0.2	18.6±0.5	9.5±0.3	26.5±0.6	26.5±0.5	15.2±0.3	35.5±0.3	25.8±0.4	16.9±0.6

A APPENDIX 1: AVERAGE FORGETTING RATE AND EXECUTION TIME

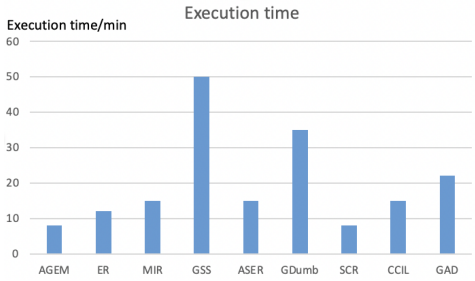

Figure 3: Execution time of all baselines and GAD on the CIFAR10 dataset.

Forgetting rate. We compute the average forgetting rate (Chaudhry et al., 2019b) of all methods. Table 3 shows that our GAD method has substantially lower forgetting rates than baselines except GDumb and SCR (in two datasets), but both GDumb and SCR's accuracy values are substantially lower than GAD (see Table 1).

The average forgetting rate is computed as follows (Chaudhry et al., 2019b): after training the model from task 1 to task j, we denote $acc_{j,i}$ as the accuracy of the trained model evaluated on the held-out test set of task $i \leq j$. The average forgetting rate FR_t at task t is:

$$FR_t = \frac{\sum_{i=1}^{t-1} f_i^t}{t-1}, \text{where} f_i^t = \max_{l \in \{1,2,\ldots,t-1\}} (acc_{l,i} - acc_{t,i}) \tag{12}$$

Figure 3 shows the training time of all systems on the CIFAR10 dataset. We can observe that GAD is faster than GSS and GDumb but slower than others. However, our GAD system achieves markedly better accuracy than all baseline systems (see Table 1).