CafeBoost: Causal Feature Boost to Eliminate Task-Induced Bias for Class Incremental Learning

Benliu Qiu Hongliang Li* Haitao Wen Heqian Qiu
Lanxiao Wang* Fanman Meng Qingbo Wu Lili Pan
University of Electronic Science and Technology of China, Chengdu, China
{qbenliu, haitaowen, hqqiu, lanxiao.wang}@std.uestc.edu.cn
{hlli, fmmeng, qbwu}@uestc.edu.cn panlili8255@gmail.com

Abstract

Continual learning requires a model to incrementally learn a sequence of tasks and aims to predict well on all the learned tasks so far, which notoriously suffers from the catastrophic forgetting problem. In this paper, we find a new type of bias appearing in continual learning, coined as task-induced bias. We place continual learning into a causal framework, based on which we find the task-induced bias is reduced naturally by two underlying mechanisms in task and domain incremental learning. However, these mechanisms do not exist in class incremental learning (CIL), in which each task contains a unique subset of classes. To eliminate the task-induced bias in CIL, we devise a causal intervention operation so as to cut off the causal path that causes the task-induced bias, and then implement it as a causal debias module that transforms biased features into unbiased ones. In addition, we propose a training pipeline to incorporate the novel module into existing methods and jointly optimize the entire architecture. Our overall approach does not rely on data replay, and is simple and convenient to plug into existing methods. Extensive empirical study on CIFAR-100 and ImageNet shows that our approach can improve accuracy and reduce forgetting of well-established methods by a large margin.

1. Introduction

Deep learning has been widely applied in many fields like computer vision, social media analytics, natural language processing, etc. The standard paradigm of deep learning is to train models on a prepared dataset, of which all data are accessible in the whole training stage. However, in most real-world applications, a sequence of tasks arrives incrementally, which requires models to learn continuously from a new task, namely continual learning [27]. Despite fine-tuning achieves this goal somewhat, such *naïve* method forgets obtained knowledge of previous tasks after learning a new task, dubbed as *catastrophic forgetting* [18].

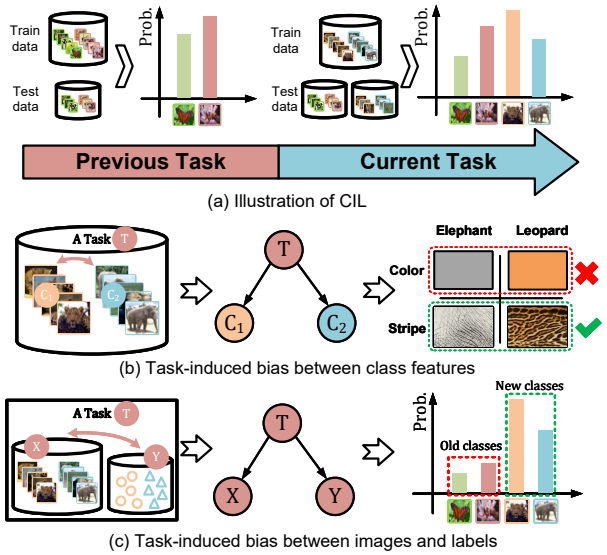

Figure 1. Illustration of CIL, task-induced bias and their detrimental effects. (a) Illustration of CIL. (b) Task-induced bias between class features. Two classes in a task are correlated by a task identifier. (c) Task-induced bias between images and labels. Images and labels in a task are correlated by a task identifier. In (b)(c), the first column describes perspectives of task-induced bias, the second shows the underlying causal effect causing task-induced bias, and the last illustrates adverse influence of task-induced bias.

Much effort has been devoted as a remedy for alleviating this problem in literature [2, 3, 11–14, 16, 33]. One general direction is to preserve knowledge representations of previous tasks by knowledge distillation [10] and by storing a few observed exemplars for replay [22]. [11] discovers the class imbalance between old classes and new ones is a key challenge of large-scale applications in class incremental learning (CIL), in which classes of a new task never appear in the previous tasks, as illustrated in Fig. 1 (a). [2, 7, 31, 35] all follow this direction and propose effective methods to overcome the class imbalance. However, these methods ignore bias problems in continual learning. In this work, we

*Corresponding authors.

consider the problem of CIL in a novel perspective of bias. We think that catastrophic forgetting is the reflection of a new-type bias, coined as task-induced bias.

Deep neural networks are easily misled by language bias [1] and vision bias [6, 9, 29], which are both derived from spurious correlations of two or more items in different positions of a sentence or an image. In contrast to these two bias, the task-induced bias occurs across different images in a common task. In CIL, data of specific classes are accessible only in a certain task if replay buffer is not applied. A deep neural network is capable of learning correlations among images of different categories or between images and labels as possible as it can, which includes a few spurious correlations, namely biases, leading to a limited generalization. On the one hand, a portion of these image correlations can be shared beneficially in this task, but these benefits cannot share among different tasks. For example, in Fig. 1 (b), color is sufficient to discriminate elephants and leopards in a specific task since elephants are grey while leopards are orange, and thus deep models have no motivation to capture more complex characteristic, like stripe. This is detrimental to the feature extractor because it captures a few details that cannot generalize among all tasks. On the other hand, as shown in Fig. 1 (c), a classifier will be biased to new classes in the current task [11, 31, 35], because the classes in the training data are all the new classes and to predict new classes is more possible to be correct.

To better understand the task-induced bias in continual learning, we model three continual learning scenarios [27] into a common causal inference framework, which merely contains three necessary variables: image, label, and task identifier. Thanks to the framework, we discover that the task-induced bias is minor in task incremental and domain incremental learning due to innate mechanisms that cut off an inappropriate causal path, which partially explain why the two scenarios are easier to tackle than CIL [27]. For the existence of the causal path in CIL, we seek an active causal intervention to close the path. To implement the intervention, we devise a simple but effective causal debias module, which is established on attention mechanisms [28]. Subsequently, we incorporate this module into existing CIL architectures and propose a training pipeline to optimize it. Our overall approach transforms causally biased features into unbiased ones, and hence we refer to it as **Ca**sual **Fe**ature **Boost** (CafeBoost). Finally, we systematically compare accuracy and forgetting of well-established methods with ours in various settings. Experimental results show that CafeBoost yields significant and consistent performance boost for existing methods in CIFAR-100 and ImageNet by a large margin (0.67%$\sim$ 12.08% on accuracy).

2. Related Work

Continual learning. Existing CIL works mostly aim to maintain knowledge representations of previous tasks, including replay-based methods and distillation-based methods. Replay-based methods [22, 25, 26] store a few examples to represent previous data and replay the stored data in the current training stage. The purpose of relay-based methods is to select the most representative samples of the entire training dataset. iCaRL [22] uses *herding strategy* [30] to choose typical exemplars for future training. [25] calculates example influence of stability and plasticity for every example in validation sets, based on which it selects examples for replay. Distillation-based methods [3, 11, 23, 24] focus on transferring learned knowledge of a previous model to the current one. LUCIR [11] combines cosine normalization, less-forget constraint, and inter-class separation to mitigate the forgetting caused by class imbalance. GeoDL [24] utilizes geodesic path for knowledge distillation which considers the gradual change between subspaces of tasks. CSCCT [3] encourages features of each class to locate in an appropriate position of the feature space, and utilizes inter-class relationships to transfer inter-class knowledge positively or negatively. CwD [23] scatters uniformly representations of each class in the initial task, which significantly benefits performance in subsequent tasks. These two types of methods are usually incorporated together to achieve better performance.

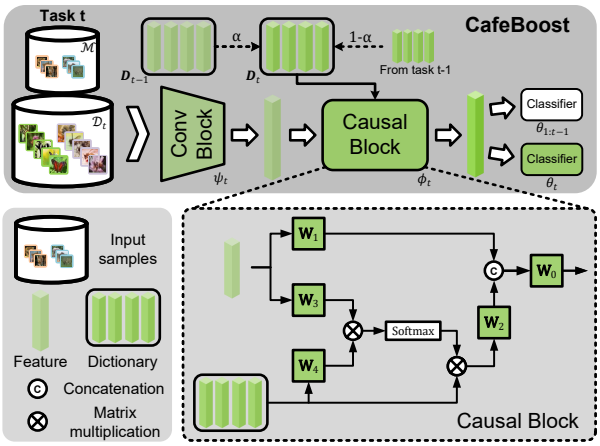

Figure 2. Overview of Casual Feature Boost (CafeBoost). Green blocks are learnable, whereas white blocks are fixed. Conv block is updated with a smaller learning rate than Causal block. The training objectives including distillation loss, margin ranking loss, etc., are omitted in this figure, since they depends on baselines. Whether to use replay buffer $\mathcal{M}$ depends on specific baselines.

Spurious correlation. Spurious correlation is also known as bias. [19] has extensively investigated the bias problem in machine learning. Many methods [1, 9, 29] have been proposed to eliminate bias in their specific applications. [1] introduces counterfactual in the training to eliminate bias in visual question answering datasets. [9] proposes a new Equalizer model to force a model look at a person instead of inferring a person's gender by context cues. [29] applies causal intervention to alleviate vision bias

among detected objects in an image.

Causal inference. Causal inference [21] aims to establish reasonable causal graphs for specific applications, then analyze problems and propose solutions based on the established causal graph. Causal intervention and counterfactual process are two powerful tools of causal inference. These two tools have been ubiquitously applied in multiple applications, such as out-of-distribution recognition [17], visual question answering [1], image captioning [34], knowledge distillation [6], etc. Recently, [12] builds a causal view to analyze the cause of forgetting in class incremental learning, and designs an anti-forgetting technique so as to distill causal effect of data without replay.

3. Preliminary

3.1. Continual learning scenarios

Continual learning can be categorized into three scenarios [27] in the light of whether a task identifier is given in a test phase and whether a model is required to infer which task it meets. If the task identifier is available during both training and test phases, this scenario is task incremental. If the task identifier is accessible in training phase but it is only necessary for a model to solve the task at hand, this scenario is domain incremental. Similar to the domain incremental scenario, the class incremental scenario doesn't provide task identifiers in test phase, but it requires a model to infer the current task identifier. Generally speaking, the class incremental scenario is the most difficult among the three, whereas the task incremental scenario is the easiest [27].

To formulate continual learning, we assume a sequence of T tasks is given as $\{\mathcal{T}_0, \mathcal{T}_1, ..., \mathcal{T}_{T-1}\}$, each with data $\mathcal{D}_t, t = 0, 1, ..., T-1$. Every sample in $\mathcal{D}_t$ is a tuple of an image $\mathbf{x}_i$ and a corresponding label y_i, and hence we can denote the data of task t as $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)_{i=1}^{N_t}\}$ where N_t is the number of samples, $\mathbf{x}_i \in \mathcal{X}_t$ and $y_i \in \mathcal{Y}_t$. Replay-based continual learning methods require a memory buffer $\mathcal{M}$ to store partial data of previous tasks. For the task $\mathcal{T}_0$, only data $\mathcal{D}_0$ can be used for training, but for the task $\mathcal{T}_t, t > 0$, we can train models using data $\mathcal{M} \cup \mathcal{D}_t$. For the task incremental scenario, task identifier $\mathcal{T}$ is given as an input to models in test phase but not for the other two scenarios.

After the above formulation, we elucidate distinctions of three incremental scenarios in a formal style [4]. Let $P(\cdot)$ denote a marginal distribution. The common ground of three scenarios is $P(\mathcal{X}_t) \neq P(\mathcal{X}_{t+1})$ for all t. Their distinctions lie in label set $\mathcal{Y}_t$ and its marginal distribution $P(\mathcal{Y}_t)$. For task incremental learning, $\mathcal{Y}_t \neq \mathcal{Y}_{t+1}$. For domain incremental learning, $\mathcal{Y}_t = \mathcal{Y}_{t+1}$ and $P(\mathcal{Y}_t) = P(\mathcal{Y}_{t+1})$. These two scenarios have no requirement on inferring task identifiers. Conversely, the class incremental learning that defines $\mathcal{Y}_t \subset \mathcal{Y}_{t+1}$ and $P(\mathcal{Y}_t) \neq P(\mathcal{Y}_{t+1})$, requires models to infer which label set is used for an arbitrary task.

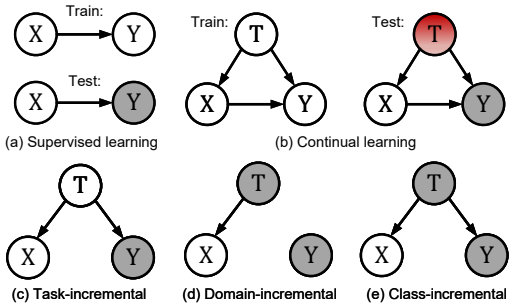

Figure 3. Causal graphs of supervised learning and continual learning. Gray nodes means unconditioned variables. (a) Causal graphs of supervised learning in training and test phase. (b) Causal graphs of continual learning. The red task variable depends on specific continual scenarios. (c) Causal path between images and labels in task incremental learning. The path $X \leftarrow T \rightarrow Y$ is impeded because variable T is conditioned. (d) Causal path between images and labels in domain incremental learning. The path $X \leftarrow T \rightarrow Y$ is cut off because of $T \nrightarrow Y$. (e) Causal path between images and labels in class incremental learning. The path $X \leftarrow T \rightarrow Y$ causes task-induced bias.

3.2. Causal graphs in continual learning

A causal graph is a directed acyclic graph, of which nodes denote variables and directed edges denote the causality between two variables. We treat images, labels and task identifiers as nodes in our causal graphs. Although only three nodes are incorporated into our causal graph, it is powerful to reveal underlying characteristics of continual learning as we will show in Sec. 4.

We denote an image as X, a label as Y and a task identifier as T in Fig. 3. The edges are listed below: $X \rightarrow Y$ means label Y is deduced from image X; $T \rightarrow X$ denotes that task identifier T have a causal influence to image X; $T \rightarrow Y$ denotes task identifier T causally affect label Y.

For supervised learning in Fig. 3 (a), its causal graph only contains an image X, a label Y and an causal path $X \rightarrow Y$ that represents a model to capture the causality between two variables. In training phase of supervised learning, both image X and label Y are accessible and we utilize them to learn the causal effect into the model. But in test phase, an image X and the causal path are known and Y is the label to be inferred. For three continual learning scenarios, images, labels and task identifiers are all given in training phase. As a result, they have the same causal graph for training in Fig. 3 (b). We illustrate their distinctions in test phase in Fig. 3 (b). Due to the involvement of the task identifier T, X may have an indirect effect on Y through the causal path $X \leftarrow T \rightarrow Y$, solely dependent on T. We will explain it in Sec. 4. See Appendix for additional explanation of causal graphs.

4. Methodology

In this section, we introduce the task-induced bias through the causal framework, a causal intervention for

CIL, its corresponding implementation, and the entire pipeline of our approach.

4.1. Task-induced bias in three continual scenarios

According to the causal graphs, we can explain why task incremental learning is the most effortless scenario among the three and why class incremental learning is the most challenging. We think that this phenomenon is caused by task-induced bias.

In Fig. 3 (b), the causal path $X \leftarrow T \rightarrow Y$ is of a fork structure, in which two kid nodes are independent conditioned on the parent. [21] points out that if T is unconditioned, the causal influence of X to Y exists, causing the task-induced bias. We give a simple example to help understand this statement. A causal graph $lighter \leftarrow smoking \rightarrow cancer$ means that a smoking person has a possibility to bring a lighter and get a lung cancer. Whether a person brings a lighter is usually independent of getting a lung cancer. Nevertheless, a smoker is highly possible to bring a lighter and to get a lung cancer, which leads to observe a correlation between $lighter$ and $cancer$. If we can control the confounder $smoking$ and collect data about non-smoking persons who usually bring lighters, then we can draw a correct conclusion that $lighter$ is independent on $cancer$.

Fortunately, mechanisms to eliminate the task-induced bias naturally exist under setups of task incremental learning and domain incremental learning. In task incremental learning shown in Fig. 3 (c), the value of parent node (task identifier) is given, hence the indirected effect of X on Y is 0 and the task-induced bias between X and Y disappears due to the characteristic of the fork structure. In domain incremental learning, the label set remains the same along task identifiers, which is implied in $\mathcal{Y}_t = \mathcal{Y}_{t+1}$ and $P(\mathcal{Y}_t) = P(\mathcal{Y}_{t+1})$. For this reason, the causal edge $T \rightarrow Y$ is cut as shown in Fig. 3 (d), resulting in disappearance of the task-induced bias. However, in the CIL, the causal path $X \leftarrow T \rightarrow Y$ cannot be cut by the task identifiers since they are inaccessible, or by the independence between T and Y like domain incremental learning. In order to eliminate the task-induced bias in CIL, we seek advanced causal techniques among which causal intervention [21] is a powerful one. It is noteworthy that our analysis indicates that a key objective of CIL is to cut the causal path $X \leftarrow T \rightarrow Y$, and we can cut it not only by conditioning variable T, but also by cutting $X \leftarrow T$ or $T \rightarrow Y$.

4.2. Causal debias

The overall causal graph of CIL is illustrated in Fig. 4 (a), which considers two facets of task-induced bias in Fig. 1. The causal effect of X_1 and X_2 on Y in the original causal graph can be formulated by Bayes' rule:

$$P(Y|X_1, X_2) = \sum_t P(Y|\underline{X_1, X_2}, t)P(t|\underline{X_1, X_2}) \quad (1)$$

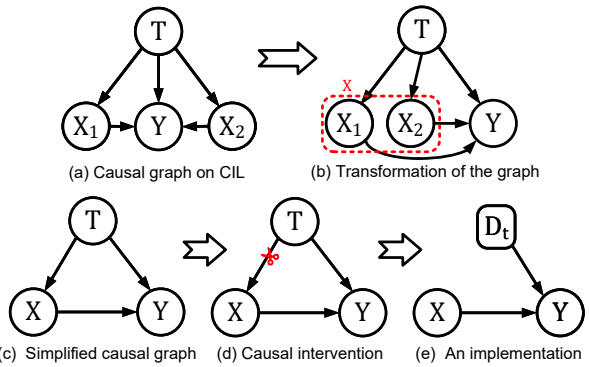

Figure 4. Causal intervention for class incremental learning. To approximate the task identifier, we utilize a class feature dictionary $\boldsymbol{D}_t$ recording information about tasks.

We can treat the two variables X_1 and X_2 as one as shown in Fig. 4 (b), because they have the same context in the causal graph. As a result, we obtain a simplified causal graph in Fig. 4 (c), of which the causal effect of X on Y is reformulated as:

$$P(Y|X) = \sum_t P(Y|X, t)P(t|X). \quad (2)$$

We have shown in Fig. 3 (e) that the causal path $X \leftarrow T \rightarrow Y$ is the cause of task-induced bias. If we can cut this causal path, the task-induced bias will be eliminated. We opt to cut the $X \leftarrow T$ using the causal intervention, as illustrated in Fig. 4 (d). The causal path $X \leftarrow T$ is introduced through the term $P(t|X)$ in Eq. 2. Following [21], the direct causal effect of X on Y is denoted by $P(Y|do(X))$, where $do(X)$ means a causal intervention on the single variable X. Because the variable T blocks all the backdoor path in the causal graph, we can obtain the expression of $P(Y|do(X))$ according to the backdoor criterion [21] as:

$$P(Y|do(X)) = \sum_t P(Y|X, t)P(t) \quad (3)$$

Compared with the old term $P(t|X)$, the new term $P(t)$ is no longer changed with X, equating to cut the causal edge $X \leftarrow T$, as shown in Fig. 4 (d).

Based on the intervened causal graph in Fig. 4 (d) in CIL, we devise a causal debias module to implement the Equation 3. Since the task identifier is not accessible in the test phase, we propose to use a feature dictionary $\boldsymbol{D}_t$ that contains information about tasks to approximate the unobserved confounder T as shown in Fig. 4 (e). The task information is retained in class labels of images for CIL, because class label sets of different tasks cannot overlap with each other and thus task identifiers can be inferred from classes. Specifically, given an image $\mathbf{x}$, a label y and a task identifier

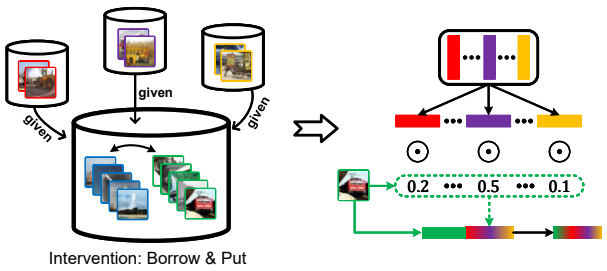

Figure 5. Illustration of causal intervention operation. This operation borrows features of classes in previous tasks, and puts them to features of new classes in the current task. The causal intervention is implied in the "borrow" and "put" operations, because the two operations mean that the features containing task information are directly controlled by ourself.

t, Equation 3 is implemented as

$$\begin{aligned} P(Y|do(X)) &= \sum_t P(y|\mathbf{x}, t)P(t) \\ &= \sum_{\boldsymbol{d}_t} P(y|\boldsymbol{x}, \boldsymbol{d}_t)P(\boldsymbol{d}_t), \end{aligned} \tag{4}$$

where $\boldsymbol{x}$ is the feature of the image $\mathbf{x}$, $\boldsymbol{d}_t$ is sampled from $\boldsymbol{D}_t$. If the last layer of a classifier is a softmax layer, we get $P(y|\boldsymbol{x}, \boldsymbol{d}_t) = \text{Softmax}(f_y(\boldsymbol{x}, \boldsymbol{d}_t))$, where $f_y(\cdot)$ is a similarity function. In a nutshell, the implementation of Equation 3 is newly formulated as:

$$P(Y|do(X)) := \mathbb{E}_{\boldsymbol{d}_t}[\text{Softmax}(f_y(\boldsymbol{x}, \boldsymbol{d}_t))]. \tag{5}$$

The above expectation can be approximated by NWGM [32] as follows:

$$\mathbb{E}_{\boldsymbol{d}_t}[\text{Softmax}(f_y(\boldsymbol{x}, \boldsymbol{d}_t))] \approx \text{Softmax}(\mathbb{E}_{\boldsymbol{d}_t}[f_y(\boldsymbol{x}, \boldsymbol{d}_t)]). \tag{6}$$

To implement our causal intervention operation in Fig. 5, we devise a simple attention-based mechanism with ignorable number of parameters:

$$f_y(\boldsymbol{x}, \boldsymbol{d}_t) = \boldsymbol{W_0}^\top \text{Cat}(\boldsymbol{W_1}^\top \boldsymbol{x}, \boldsymbol{W_2}^\top g_y(\boldsymbol{d}_t)), \tag{7}$$

where $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ both are fully connected layers and $\text{Cat}(\cdot)$ is the concatenation operator. Due to the additive property of expectation, the $\mathbb{E}_{\boldsymbol{d}_t}[f_y(\boldsymbol{x}, \boldsymbol{d}_t)]$ can be rewritten as:

$$\mathbb{E}_{\boldsymbol{d}_t}[f_y(\boldsymbol{x}, \boldsymbol{d}_t)] = \boldsymbol{W_0}^\top \text{Cat}(\boldsymbol{W_1}^\top \boldsymbol{x}, \boldsymbol{W_2}^\top \mathbb{E}_{\boldsymbol{d}_t}[g_y(\boldsymbol{d}_t)]). \tag{8}$$

We devise $\mathbb{E}_{\boldsymbol{d}_t}[g_y(\boldsymbol{d}_t)]$ utilizing a scaled dot-product attention [28], which is calculated as follows:

$$\begin{aligned} &\boldsymbol{Q} = \boldsymbol{W_3}^\top \boldsymbol{x}, \boldsymbol{K} = \boldsymbol{W_4}^\top \boldsymbol{D}_t, \boldsymbol{V} = \boldsymbol{D}_t, \\ &\boldsymbol{A} = \text{Softmax}(\boldsymbol{Q}^\top \boldsymbol{K}/\sqrt{\sigma}), \\ &\mathbb{E}_{\boldsymbol{d}_t}[g_y(\boldsymbol{d}_t)] = \boldsymbol{V}^\top \boldsymbol{A}, \end{aligned} \tag{9}$$

where $\boldsymbol{W_3}$ and $\boldsymbol{W_4}$ both are fully connected layers, and σ is the output dimension of them.

Algorithm 1 CafeBoost training pipeline.

Input: : Training data $\mathcal{D}_0, \cdots, \mathcal{D}_T$, empty dictionary $\boldsymbol{D}_0$, empty example memory $\mathcal{M}$,
Initialize parameters of feature extractor $\boldsymbol{\psi}_0$, causal debias module $\boldsymbol{\phi}_0$ (Eq. 8, 9) and classifier $\boldsymbol{\theta}_0$
// First task training stage.
Train $\boldsymbol{\psi}_0, \boldsymbol{\theta}_0$ on $\mathcal{D}_0$ by minimizing $\mathcal{L}_{CE}$
Extract example features on $\mathcal{D}_0$, using $\boldsymbol{\psi}_0$
Store class mean features on $\mathcal{D}_0$ to dictionary $\boldsymbol{D}_0$
// Causal debias module training stage.
Train $\boldsymbol{\phi}_0, \boldsymbol{\theta}_0$ on $\mathcal{D}_0$ by minimizing $\mathcal{L}_{CE}$
Extract example features on $\mathcal{D}_0$, using $\boldsymbol{\psi}_0$ and $\boldsymbol{\phi}_0$
if example memory $\mathcal{M}$ is used **then**
  Update example memory $\mathcal{M}$
end if
// Class incremental training stage.
for t in $\{1, ..., T\}$ **do**
  Add a new classifier $\boldsymbol{\theta}_t$
  Train $\boldsymbol{\psi}_t$, $\boldsymbol{\phi}_t$ and $\boldsymbol{\theta}_t$ on $\mathcal{D}_t \cup \mathcal{M}$ (or $\mathcal{D}_t$) by Eq. 10
  Extract example features on $\mathcal{D}_t$, using $\boldsymbol{\psi}_t$ and $\boldsymbol{\phi}_t$
  if example memory $\mathcal{M}$ is used **then**
    Update example memory $\mathcal{M}$
    Update dictionary $\boldsymbol{D}_t$ by Eq. 11
  end if
end for

Output: Final feature extractor $\boldsymbol{\psi}_T$, final causal debias module $\boldsymbol{\phi}_T$, and all classifiers $\{\boldsymbol{\theta}_0, ..., \boldsymbol{\theta}_T\}$

4.3. Causal Feature Boost

In this section, we describe how the causal debias module is added into the continual learning pipeline, and the training strategies to enable the overall architecture. Additionally, we propose a momentum method to update the dictionary so as to catch more task information.

Architecture. Our causal debias module can be added between the feature extractor and the classifier of existing continual learning architectures. In Fig. 2, given an image as input, the CNN backbone (e.g. ResNet [8]) will extract a feature map. Then, we feed this feature map and a pre-collected dictionary into our causal debias module, and achieve a newly-updated feature map which is forwarded to the classifier to get a class label. After a new task comes, we add one more classifier for new classes and add no other layers.

Training pipeline. Because the dictionary used in the causal debias module need to be filled with average features of classes, we train a feature extractor and a classifier by using data of the initial phase firstly. Then, we fix the feature extractor and run an extra phase before the incremental phases to train the newly-added causal debias module and the classifier together. In the remaining phases,

Method	CIFAR-100			ImageNet-Sub			ImageNet-Full	
Average accuracy(%)	5	10	25	5	10	25	5	10
LwF† [16]	49.59	46.98	45.51	53.62	47.64	44.32	44.35	38.90
BiC† [31]	59.36	54.20	50.00	70.07	64.96	57.73	62.65	58.72
iCaRL [22]	62.95	56.92	51.30	65.04	60.82	54.56	51.50	46.89
LUCIR [11]	63.56	57.20	50.29	70.84	68.32	61.44	64.45	61.57
GeoDL [24]	67.08	60.96	53.98	71.27	**70.24**	64.86	65.23	64.46
CwD [23]	66.84	62.66	56.28	67.36	64.09	60.52	-	-
CSCCT [3]	66.89	59.91	52.32	71.33	67.91	61.52	-	-
iCaRL+CafeBoost	69.41	64.70	61.60	70.53	67.04	63.40	53.46	48.46
LUCIR+CafeBoost	**70.10**	**66.56**	**62.37**	**71.51**	70.03	**67.74**	**66.88**	**64.77**
Forgetting rate(%)	5	10	25	5	10	25	5	10
LwF† [16]	43.36	43.58	41.66	55.32	57.00	55.12	48.70	47.94
BiC† [31]	31.42	32.50	34.60	27.04	31.04	37.88	25.06	28.34
iCaRL [22]	22.95	25.30	25.74	18.70	21.35	25.09	26.03	33.76
LUCIR [11]	23.30	26.40	29.21	31.88	33.48	35.40	24.08	27.29
GeoDL [24]	16.83	21.83	24.91	8.68	9.07	11.97	11.03	12.81
CwD [23]	17.70	26.80	26.91	11.16	17.53	25.41	-	-
iCaRL+CafeBoost	8.51	12.91	16.08	12.20	14.23	18.52	25.19	28.24
LUCIR+CafeBoost	**6.99**	**9.03**	**11.75**	**8.16**	**7.50**	**9.10**	**8.65**	**11.73**

Table 1. The average incremental accuracy and the forgetting rate on CIFAR-100, ImageNet-Sub and ImageNet-Full. The number of incremental phases are 5, 10, and 25. The number of exemplars per class is 20. An ideal continual learning method should have high average incremental accuracy and low forgetting rate. Results of models with † are taken from [24].

the causal debias module are always learnable, whereas the classifiers of old tasks are fixed. Our causal debias module can be used in iCaRL [22] and LUCIR [11] as a plugin. Our overall approach borrows their corresponding training objectives. Therefore, the training objective of our approach (CafeBoost) is formulated as:

$$\min_{\theta}[L_{ce}(x, y, \theta) + \eta L_{*}(\theta)], \tag{10}$$

where $L_{*}(\theta)$ denotes specific losses of the corresponding base methods, e.g., distillation loss for iCaRL, margin ranking loss and others for LUCIR. Algorithm 1 details out the CafeBoost training pipeline.

Momentum dictionary updating. Our approach exploits an external dictionary to store mean features of each class in the initial phase. We can exploit a momentum updating strategy to renew features in the dictionary, if replay buffer is used. The new dictionary at task t is updated as

$$\boldsymbol{D}_t = \alpha \boldsymbol{D}_{t-1} + (1-\alpha)[\mathbf{f}_1^{t-1}, ..., \mathbf{f}_c^{t-1}, ..., \mathbf{f}_C^{t-1}], \tag{11}$$

where α controls the updating rate, $\mathbf{f}_c$ represents a mean feature of class c, and C is the number of classes.

5. Experiments

5.1. Settings

Datasets & Protocols. We evaluate our approach on two commonly-used datasets: CIFAR-100 [15] and ImageNet [5]. CIFAR-100 contains 100 classes and each class has 500 images for training and 100 images for test, with each image of size 32×32. ImageNet with about 1.3 million images of size 224×224 is divided into two CIL settings: ImageNet-Sub that owns 100 classes and ImageNet-Full that owns 1000 classes. We follow the protocol used in many recent works [11,23,24]. Models are firstly trained on half classes, i.e., 50 classes for CIFAR100 and ImageNet-Sub, and 500 for ImageNet-Full. Subsequently, we start incremental learning phases with adding a fixed number of unobserved classes in each phase. We choose three different numbers of incremental phases: 5, 10, and 25.

Evaluation Metrics. The two commonly-used metrics to evaluate performance of a CIL model are 1) the average incremental accuracy [22] $\mathcal{A}_T$, and 2) the average forgetting $\mathcal{F}_T$. The average incremental accuracy and the average forgetting after T tasks are defined as

$$\mathcal{A}_T = \frac{1}{T}\sum_{t=1}^{T} A_t, \mathcal{F}_T = \frac{1}{T-1}\sum_{t=2}^{T} F_t,$$

where $a_{i,j}$ denotes the accuracy on task i after the model learning task j, $A_t = \frac{1}{t}\sum_{i=1}^{t} a_{i,t}$, and $F_t = \frac{1}{t-1}\sum_{j=2}^{t} \max_i(a_{i,t} - a_{j,t})$ with $i \in 1, ..., j-1$. We report top-1 accuracy.

Implementation details. All methods are implemented with PyTorch [20]. We use the ResNet18 for CIFAR-100 and ImageNet following [23]. For all experiments, SGD optimizer and the batch size 128 are used. For CIFAR-100 and ImageNet, we train models for 160 epochs and 90 epochs, respectively. For CIFAR-100, our results are averaged over three runs with different orders generated by random seed 1993. For ImageNet, the first order with random seed 1993

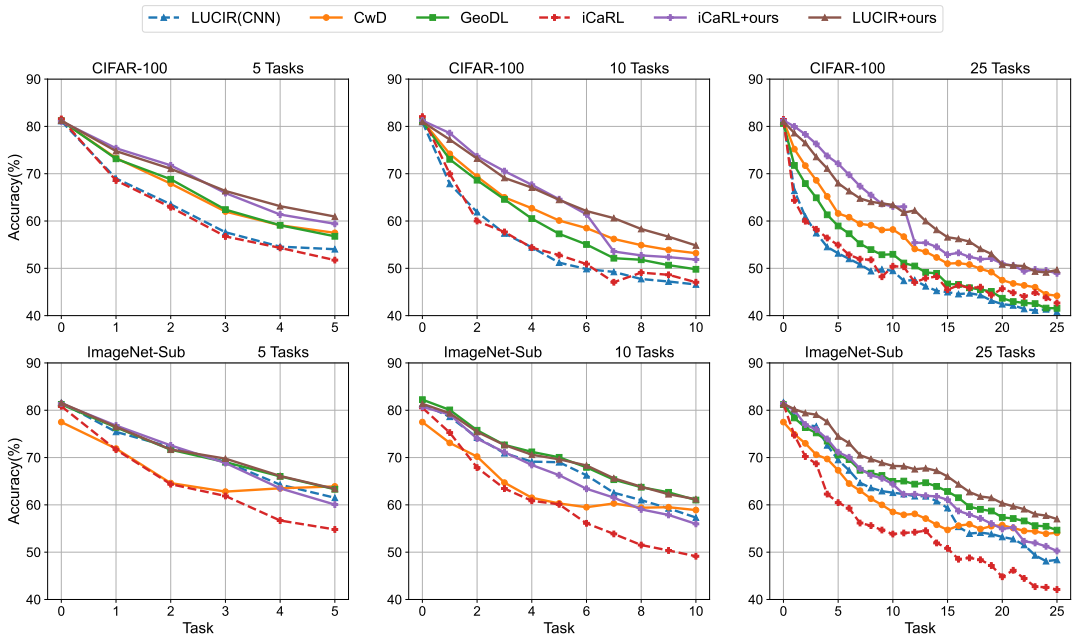

Figure 6. Results on CIFAR-100 (5, 10, and 25 tasks), ImageNet-Sub (5, 10, and 25 tasks). The horizontal axis shows the task identifier and the vertical axis is the corresponding average incremental accuracy.

is conventionally used. In addition, we deploy CafeBoost to two baselines: iCaRL [22] and LUCIR [11] and compare it with LwF [16], BiC [31], GeoDL [24], CwD [23] and CSCCT [3] on all datasets. GeoDL, CwD and CSCCT are implemented based on LUCIR.

5.2. Evaluations

Results on CIFAR-100. Tab. 1 summarizes the experimental results of CIFAR-100 in three different numbers of incremental tasks. We observe that CafeBoost improves iCaRL baseline by 6.46%, 7.78%, 10.30% for 5, 10, and 25 tasks, respectively. After adding to LUCIR baseline, we observe a performance boost: 6.54%, 9.36%, 12.08%, for 5, 10, and 25 tasks. The corresponding average incremental accuracies reach high levels of 70.10%, 66.56%, and 62.37%. Furthermore, CafeBoost achieves lower forgetting rates after applying to the baselines. It can decrease the forgetting rates of iCaRL by 14.44%, 12.39%, and 9.66% and those of LUCIR by 16.31%, 17.37%, and 17.46%, for 5, 10, and 25 tasks. The first row of Fig. 6 shows CafeBoost improves performance of baselines by a large margin, achieves higher accuracies and prevents catastrophic forgetting better than the prior methods for CIL. See Appendix for more results about the forgetting rates.

Results on ImageNet. Tab. 1 also reports experimental results of ImageNet-Sub and ImageNet-Full. On ImageNet-Sub, CafeBoost boosts average incremental accuracies by $0.67\% \sim 8.84\%$ over iCaRL and LUCIR baselines, for 5, 20, and 25 tasks, and decreases the corresponding forgetting rates by $6.57\% \sim 26.30\%$. For example, CafeBoost improves average incremental accuracy of LUCIR from 61.44% to 67.74% (+6.30%), and reduces forgetting rate from 35.40% to 9.10% (-26.30%). The second row of Fig. 6 further validates effectiveness of CafeBoost for all numbers of incremental phases. Additionally, we observe in Fig. 6 CafeBoost has almost no effect on the initial phase, but brings great boost in the following incremental phases. CafeBoost also enhances performance of iCaRL and LUCIR on ImageNet-Full, as shown in Tab. 1. We observe CafeBoost improves accuracies of iCaRL by 1.96%, 1.57%, and those of LUCIR by 2.43%, 3.20% for 5 and 10 incremental phases respectively.

S	B	10	20	30	40	50
10	LUCIR	59.12	60.26	61.77	61.62	62.91
	+Ours	55.87	62.61	66.21	68.22	69.70
	↑	-3.25	+2.35	+4.44	+6.60	+6.79
5	LUCIR	54.62	56.20	56.65	57.29	56.90
	+Ours	49.02	58.53	63.39	65.73	65.95
	↑	-5.60	+2.33	+6.74	+8.44	+9.05

Table 2. Ablation study on impact of number of classes in the initial phase. B denotes number of classes in the initial phase. S is the number of new classes per incremental step. Experiments are conducted on CIFAR-100 with 20 exemplars per class. The reported values (%) are average incremental accuracies. ↑ denotes the increment of values.

5.3. Ablation Studies

In this section, we conduct experiments to study the effect on our approach of 1) number of classes in initial phase,

S	R	LUCIR	+Ours	↑
10	5	54.23±0.78	64.80±0.36	+10.57
	10	59.98±0.54	68.30±0.31	+8.32
	20	63.56±0.55	70.10±0.39	+6.54
5	5	47.74±0.84	62.23±0.67	+14.49
	10	53.22±1.28	64.08±0.77	+10.86
	20	57.20±1.14	66.56±0.80	+9.36

Table 3. Ablation study on impact of number of exemplars. R represents the number of exemplars per class. S is the number of new classes per incremental step. Experiments are conducted on CIFAR-100. The reported values (%) are average incremental accuracies. ↑ denotes the increment of values.

Methods	CIFAR-100		ImageNet-Sub	
	Accuracy	Forgetting	Accuracy	Forgetting
LUCIR	40.10±1.27	52.35±1.31	60.47	25.73
+Ours	59.01±0.22	23.31±0.76	65.15	18.05

Table 4. Comparisons of average incremental accuracies and forgetting rates with no replay data. Experiments are conducted with 5 incremental steps. 20 exemplars are stored per class.

2) number of exemplars in memory, 3) different classifiers, and 4) coefficient of momentum dictionary update.

Impact of number of classes in initial phase. We investigate CafeBoost with 10, 20, 30, 40, and 50 classes of initial phase in Tab. 2. CafeBoost can enhance performance when the number of classes in initial phase is larger than 10. The performance improvement increases from negative to positive, along the number for both $S = 10$ and $S = 5$. When B equates 10, our approach has a negative influence on the baseline. This is because characteristics of classes in the initial phase are insufficient to share for all classes in the dataset. However, when $B >= 20$, our approach boosts the baseline by a large margin, which shows that it tolerates a broad range of the number B.

Impact of number of exemplars in memory. We vary the number of exemplars per class and show its effect in Tab. 3. Both the performance of baselines and CafeBoost improve along the increase of exemplar number per class, which is consistently observed for $S = 10$ and $S = 5$. In addition, the improvement margin decreases along the increase of R. For instance, $+10.57$ for $S = 10$ and $R = 5$, whereas $+6.54$ for $S = 10$ and $R = 20$. It is worth noting that CafeBoost does not rely on a replay buffer. Tab. 4 shows CafeBoost outperforms remarkably the baseline on accuracy and forgetting in CIFAR-100 and ImageNet-Sub.

Impact of different classifiers. We also conduct experiments to research behaviors of CafeBoost combined with three different classifiers. We adopt the classifiers CNN, k-NME, and AME, as in [11]. CNN classifies examples by exploiting one class prototype per class. k-NME uses k samples of each class to represent the class, while AME utilizes all samples. Tab. 5 shows our approach achieves the best performance when using AME classifier. Moreover, it is noteworthy that our approach enhances performance of baselines using all three classifiers by a large margin in various numbers of tasks.

Method		iCaRL	iCaRL+ours	LUCIR	LUCIR+ours
CNN	5	53.47±0.15	56.53±0.26	63.56±0.55	70.10±0.39
	10	49.30±0.43	53.75±0.03	57.20±1.14	66.56±0.80
	25	45.59±0.75	51.18±0.78	50.29±1.14	62.37±1.17
k-NME	5	62.95±0.51	69.40±0.23	65.42±0.67	69.77±0.35
	10	56.91±0.64	64.70±0.40	59.85±1.20	66.53±0.72
	25	51.30±0.92	61.6±0.90	52.07±0.43	62.64±1.30
AME	5	65.02±0.64	70.61±0.29	66.76±0.84	70.54±0.31
	10	58.27±0.63	66.52±0.46	61.54±1.36	67.77±0.71
	25	52.69±0.92	64.10±0.89	54.84±1.78	65.10±1.35

Table 5. The average incremental accuracy by using different classifiers (CNN, k-NME, AME) with 20 exemplars per class. Values are evaluated on CIFAR-100 with 5, 10, and 25 incremental steps.

Sensitivity of momentum coefficient. Fig. 7 shows our approach is not sensitive to the momentum coefficient of dictionary updating for all three classifiers. We set α to 0.9 for other experiments of our approach.

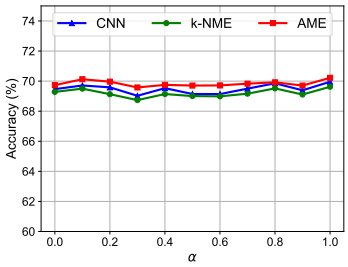

Figure 7. Experiments on sensitivity of momentum coefficient. Average accuracies are evaluated on CIFAR-100 with 5 incremental steps and 20 exemplars per class.

6. Conclusions

To the best of our knowledge, this work is the first to consider a new type of bias - task-induced bias in a causal perspective, which is derived from special settings of continual learning. We analyzed three continual scenarios in a causal framework, and found that the task-induced bias is particularly detrimental to CIL. Based on this framework, we designed a causal intervention operation and implemented it as a causal debias module by exploiting a powerful attention mechanism. Comprehensive experiments demonstrated our plugin method can boost performance of baselines and outperform several existing strong methods. In the future, we think it is a promising direction to regard the task-induced bias as a problem in CIL and design more exquisite approaches to eliminate it by cutting the corresponding causal path.

Acknowledgments. This work was supported in part by National Key R&D Program of China (2021ZD0112001) and National Natural Science Foundation of China (No.61831005, 61971095 and 62271119).

References

[1] Ehsan Abbasnejad, Damien Teney, Amin Parvaneh, Javen Shi, and Anton van den Hengel. Counterfactual vision and language learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10041—-10051, 2020. 2, 3

[2] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. SS-IL: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision*, pages 844–853, 2021. 1

[3] Arjun Ashok, K. J. Joseph, and Vineeth Balasubramanian. Class-incremental learning with cross-space clustering and controlled transfer. In *European Conference on Computer Vision*, 2022. 1, 2, 6, 7

[4] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021. 3

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6

[6] Xiang Deng and Zhongfei Zhang. Comprehensive knowledge distillation with causal intervention. In *Advances in Neural Information Processing Systems*, pages 22158–22170, 2021. 2, 3

[7] Chen He, Ruiping Wang, and Xilin Chen. A tale of two CILs: The connections between class incremental learning and class imbalanced learning, and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*, pages 3554–3564, 2021. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. 5

[9] Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. In *European Conference on Computer Vision*, 2018. 2

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *ArXiv: 1503.02531*, 2015. 1

[11] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019. 1, 2, 6, 7, 8

[12] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3956–3965, 2021. 1, 3

[13] K. J. Joseph, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Vineeth N. Balasubramanian. Energy-based latent aligner for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7452–7461, 2022. 1

[14] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16071–16080, 2022. 1

[15] Alex Kriszhevsky. Learning multiple layers of features from tiny images. In *Technical report*, 2009. 6

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 1, 6, 7

[17] Chengzhi Mao, Kevin Xia, James Wang, Hao Wang, Junfeng Yang, Elias Bareinboim, and Carl Vondrick. Causal transportability for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7521–7531, 2022. 3

[18] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, 1989. 1

[19] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, 2021. 2

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019. 6

[21] Judea Pearl, Madelyn Glymour, and Nicholas P. Jewell. *Causal Inference In Statistics: A primer*. Wiley, 2016. 3, 4

[22] Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5533–5542, 2017. 1, 2, 6, 7

[23] Yujun Shi, Kuangqi Zhou, Jian Liang, Zihang Jiang, Jiashi Feng, Philip H. S. Torr, Song Bai, and Vincent Y. F. Tan. Mimicking the oracle: An initial phase decorrelation approach for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16722–16731, 2022. 2, 6, 7

[24] Christian Simon, Piotr Koniusz, and Mehrtash Harandi. On learning the geodesic path for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1591–1600, 2021. 2, 6, 7

[25] Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. Exploring example influence in continual learning. In *Advances in Neural Information Processing Systems*, 2022. 2

[26] Rishabh Tiwari, Krishnateja Killamsetty, Rishabh Iyer, and Pradeep Shenoy. GCR: Gradient coreset based replay buffer selection for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 99–108, 2022. 2

[27] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. In *arXiv: 1904.07734*, 2019. 1, 2, 3

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 5999–6009, 2017. 2, 5

[29] Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. Visual commonsense R-CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 2

[30] Max Welling. Herding dynamical weights to learn. In *International Conference on Machine Learning*, 2009. 2

[31] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 1, 2, 6, 7

[32] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 5

[33] Shipeng Yan, Jiangwei Xie, and Xuming He. DER: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3013–3022, 2021. 1

[34] Xu Yang, Hanwang Zhang, and Jianfei Cai. Deconfounded image captioning: A causal retrospect. *Transactions on Pattern Analysis and Machine Intelligence*, 2021. 3

[35] 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*, pages 13205–13214, 2020. 1, 2