Layerwise Optimization by Gradient Decomposition for Continual Learning

Shixiang Tang[1†] Dapeng Chen[3] Jinguo Zhu[2] Shijie Yu[4] Wanli Ouyang[1]
[1]The University of Sydney, SenseTime Computer Vision Group, Australia [2]Xi'an Jiaotong University
[3]Sensetime Group Limited, Hong Kong [4]Shenzhen Institutes of Advanced Technology, CAS
tangshixiang@sensetime.com dapengchenxjtu@yahoo.com wanli.ouyang@sydney.edu.au

Abstract

Deep neural networks achieve state-of-the-art and sometimes super-human performance across various domains. However, when learning tasks sequentially, the networks easily forget the knowledge of previous tasks, known as "catastrophic forgetting". To achieve the consistencies between the old tasks and the new task, one effective solution is to modify the gradient for update. Previous methods enforce independent gradient constraints for different tasks, while we consider these gradients contain complex information, and propose to leverage inter-task information by gradient decomposition. In particular, the gradient of an old task is decomposed into a part shared by all old tasks and a part specific to that task. The gradient for update should be close to the gradient of the new task, consistent with the gradients shared by all old tasks, and orthogonal to the space spanned by the gradients specific to the old tasks. In this way, our approach encourages common knowledge consolidation without impairing the task-specific knowledge. Furthermore, the optimization is performed for the gradients of each layer separately rather than the concatenation of all gradients as in previous works. This effectively avoids the influence of the magnitude variation of the gradients in different layers. Extensive experiments validate the effectiveness of both gradient-decomposed optimization and layerwise updates. Our proposed method achieves state-of-the-art results on various benchmarks of continual learning.

1. Introduction

Recent years have witnessed the great progress in deep learning (DL) through training on large datasets. A typical supervised DL task requires independent and identically distributed (*i.i.d*) samples from a stationary distribution [15, 39, 46]. However, many DL models deployed in the real-world are exposed to non-stationary situations where data is acquired sequentially and its distribution varies over time. In this scenario, DNNs trained by Stochastic Gradient Descent (SGD) will easily forget the knowledge from previous tasks while adapting to the information from the incoming tasks. This phenomenon, known as *catastrophic forgetting*, invokes more effective algorithms for continual learning (CL), the goal of which is to learn consecutive tasks without severe performance degradation on previous tasks [5, 30, 34, 38, 44, 43, 57, 57, 50].

One of the popular attempts for continual learning relies on a set of episodic memories, where each episodic memory stores representative data from an old task [5, 38, 30]. In particular, the network parameters are jointly optimized by the recorded samples that are regularly replayed and the samples drawn from the new task. An effective solution is to modify the gradients for updates. The methods like GEM, A-GEM and S-GEM obtain gradient update by requiring the loss of each old task does not increase [7, 30]. In practice, the constraints are imposed by forcing the inner product between the gradient for update and the gradient of every old task non-negative.

In this work, we consider that the gradients of multiple tasks have mixed information and should be disentangled when used for learning. There are two parts mixed in the gradients, shared gradient and task-specific gradients. Optimization along the shared gradient will be beneficial for memorizing all old tasks, but optimization along task-specific gradients will fall into a dilemma that optimizing one episodic memory will inevitably damage another. Therefore, different constraints are imposed on two gradients. We encourage the consistency between gradient for update and the shared gradient but expect that the gradient for update will be orthogonal to all task-specific gradients. The first constraint can be mathematically presented as the inner product of shared gradient and gradient for update non-negative. The second constraint requires the gradient for update should be orthogonal to the space spanned by all task-specific gradients. Our further analysis points out that the second constraints can be relaxed by PCA, which captures the most important gradient constraints.

Furthermore, we observe the large variation of magnitudes for the gradients in different layers. However, the previous gradient modification methods ignore the intrinsic

[2]This work was done when Shixiang Tang was an intern at SenseTime.

magnitude variations of different layers. They concatenate the gradients from different layers into a vector and construct optimization problems under the constraints made by these concatenated vectors. We argue that during optimization, gradients from some layers have larger magnitudes and they will dominate the solution of the optimization problem. However, no evidence shows that these layers are more important for loss minimization than the others. To address this problem, we propose a layer-wise gradient update strategy, where unique gradient constraints are imposed by each layer for optimization and the solution is only specific to the parameters in that layer. Our further analysis manifests that the layer-wise optimization strategy increases the efficiency of reducing old task losses.

The contribution of this paper is two-fold: (1) Gradient decomposition is leveraged to specify the shared and task-specific information in the episodic memory. Different constraints are imposed based on the shared gradient and tasks-specific gradient respectively. (2) Layer-wise gradient update strategy is proposed to deal with large magnitude variations between gradients from different layers and thus it can reduce the losses of episodic memory more efficiently. Extensive ablation studies validate the effectiveness of the two improvements.

2. Related Work

2.1. Continual Learning

Continual learning has been a long-standing research problem in the field of machine learning [32, 45, 36, 47]. Generally speaking, there are three different types of scenarios for continual learning: (1) class-incremental scenario, where the number of class labels keeps growing but no explicit task boundary is defined under test; (2) task-incremental scenario, where the boundaries among tasks are assumed known and the information about the task under test is given. (3) data-incremental scenario, where the set of class labels are identical for all tasks. Existing work of above three scenarios mainly falls into two categories: Regularization-based and Memory-based approaches.

Regularization-based Approach: Regularization approaches do not require data from old tasks. Rather, they attempted to tackle catastrophic forgetting by discouraging changes on important parameters or penalizing the change of activations on old tasks. The former approaches, such as EWC [22], SI [55], MAS [1] and ALASSO [35] relied on different estimations of parameters importance, which was usually conducted by exhaustive search [42] or variational Bayesian methods [33, 56]. The latter approaches, such as LWF [29], LFL [20], LWM [9] and BLD [11], leveraged knowledge distillation [16, 58] among consecutive tasks.

Memory-based Approach: Memory-based approaches leverage episodic memory that stores representative samples from each old task to overcome catastrophic forgetting. One popular framework is multitask learning, where the model shares the same backbone but is equipped with different task-specific classifiers. During training, all episodic memory are mixed to form an old task and the loss of the model is defined by both old and new tasks. Example methods of this framework were iCarl [38], End2End [5] and DR [17]. Under this framework, various problems were considered, such as data imbalance [18, 35], using unlabeled data [26] and the bias of fully-connected layers [53]. Another popular framework for memory-based approaches is to use episodic memory tasks to construct the optimization problems. GEM [30] was the first work under this framework. In the method, each episodic memory constructed one constraint independently. During training the new task, GEM ensured the loss of every episodic memory non-increase. Further improvements on GEM such as A-GEM [7] and S-GEM [7] relaxed the constraints by either considering the loss averaged on all old tasks or constraining the loss of a random episodic memory every training iteration. Although these methods improved GEM in different aspects, they treated every episodic memory independently and no further analysis or decomposition of these gradients was considered in these works.

2.2. Layerwise Gradient Update

Stochastic Gradient Descent is the most widely used optimization techniques for training DNNs [3, 31, 2]. However, it applied the same hyper-parameters to update all parameters in different layers, which may not be optimal for loss minimization. Therefore, layerwise adaptive optimization algorithms were proposed[10, 21]. RMSProp [41] altered the learning rate of each layer by dividing the square root of its exponential moving average. LARS [54] let the layerwise learning rate be proportional to the ratio of the norm of the weights to the norm of the gradients. Both layerwise adaptive optimizers solved the variation of update frequencies for different layers and thus outperformed SGD in various large-scale benchmarks. Layerwise gradient update strategy is also applied in meta-learning for rapid loss convergence on transfer learning [13, 14, 48] and few-shot learning [49, 28]. [12] proposed WarpGrad which layerwisely meta-learned to warp task loss surfaces across the joint task-parameter distribution to facilitate gradient descent. MT-Net [27] enabled the meta-learner to learn on each layer's activation space, a subspace that the task-specific learner performed gradient descent on. Our method differs from the above methods in two aspects. Our layerwise gradient update strategy aims at preserving old knowledge. Instead, RMSProp, LARS aimed to learn new tasks more efficiently, and WarpGrad, MT-Net aimed to transfer more related information in source data to target samples, which have different targets with our method. Secondly, RMSProp and LARS tried to handle the different update frequency between parameters in different layers but our

method aims at handling large magnitude variation of parameters in different layers.

3. Methodology

For continual learning, we first define a task sequence $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_N\}$ of N tasks. For the t-th task $\mathcal{T}_t$, there is a training dataset $\mathcal{D}_t$ and a memory coreset $\mathcal{M}_t^{\text{cor}}$. In particular, $\mathcal{D}_t = \{(x_t^i, y_t^i)\}_{i=1}^{n_t}$, where each instance (x_t^i, y_t^i) is composed of an image $x_t^i \in \mathcal{X}_t$ and a label $y_t^i \in \mathcal{Y}_t$. The coreset is represented by $\mathcal{M}_t^{\text{cor}} = \mathcal{M}_{t-1}^{\text{cor}} \cup \mathcal{M}_t$, where $\mathcal{M}_t$ is the episodic memory of task $\mathcal{T}_t$ that stores representative data from $\mathcal{D}_t$.

The goal at the t-th step is to train a function f to perform the current task $\mathcal{T}_t$ as well as the previous tasks $\mathcal{T}_{1:(t-1)} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_{t-1}\}$. During training, all data in $\mathcal{D}_t$ and $\mathcal{M}_{t-1}^{\text{cor}}$ are used. Considering f is parameterized by θ_t, we define the losses for new task $\mathcal{T}_t$ and any previous task $\mathcal{T}_i (i < t)$ using the training data $\mathcal{D}_t$ and episodic memory $\mathcal{M}_i$ respectively, i.e.,

$$\begin{aligned} \mathcal{L}^{\text{new}}(\theta_t, \mathcal{D}_t) &= \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} CE(f(x;\theta_t), y) \\ \mathcal{L}^{\text{old}}(\theta_t, \mathcal{M}_i) &= \frac{1}{|\mathcal{M}_i|} \sum_{(x,y)\in\mathcal{M}_i} CE(f(x;\theta_t), y). \end{aligned} \tag{1}$$

where CE is the standard cross-entropy loss. The gradient of new task is defined as:

$$g = \frac{\partial \mathcal{L}^{\text{new}}(\theta_t, \mathcal{D}_t)}{\partial \theta_t}. \tag{2}$$

3.1. Gradient Decomposition

Biologically, continual learning is inspired by human learning. We learn the new tasks by applying the knowledge previously learned from the related tasks. In this way, there exists common knowledge shared among the old tasks. Given the losses of old task $\{\mathcal{L}^{\text{old}}(\theta_t, \mathcal{M}_i)\}_{i=1}^{t-1}$, we assume each loss has two components. One is the shared loss, which is the same for any previous task. Minimizing the shared loss would improve the overall performance across all previous tasks. The other is task-specific component, reflecting the task-specific knowledge of every old task. For $\mathcal{L}^{old}(\theta_t, \mathcal{M}_i)$ about the i-th episodic memory, it can be written as:

$$\mathcal{L}^{old}(\theta_t, \mathcal{M}_i) = \mathcal{L}^{old}_{\text{shared}}(\theta_t, \mathcal{M}^{cor}_{t-1}) + \mathcal{R}^{old}_{\text{specific}}(\theta_t, \mathcal{M}_i) \tag{3}$$

where $\mathcal{L}^{old}_{\text{shared}}(\theta^t, \mathcal{M}^{cor}_{t-1})$ is the shared loss term, which is the same for different old tasks, and $\mathcal{R}^{old}_{\text{specific}}(\theta_t, \mathcal{M}_i)$ is the residual for the task $\mathcal{T}_i$.

Shared Gradient. The shared gradient is driven by $\mathcal{L}^{old}_{\text{shared}}(\theta^t, \mathcal{M}^{cor}_{t-1})$. We define the shared loss by averaging all losses in the memory coreset $\mathcal{L}^{old}_{\text{shared}}(\theta^t, \mathcal{M}^{cor}_{t-1})$ $= \frac{1}{t-1}\sum_{i=1}^{t-1} \mathcal{L}^{old}(\theta_t, \mathcal{M}_i)$. Thus, the shared gradient $\bar{g}$ can be obtained as follows:

$$\bar{g} = \frac{\partial \mathcal{L}^{old}_{\text{shared}}(\theta^t, \mathcal{M}^{cor}_{t-1})}{\partial \theta_t} = \frac{1}{t-1}\sum_{i=1}^{t-1} \frac{\partial \mathcal{L}^{\text{old}}(\theta_t, \mathcal{M}_i)}{\partial \theta_t}, \tag{4}$$

where the equality holds since partial derivatives are linear.

Task-specific Gradients. The task-specific gradients $\hat{g}_i$ are then obtained by subtracting the shared gradient from the gradient of each task,

$$\hat{g}_i = \frac{\partial \mathcal{L}^{\text{old}}(\theta_t, \mathcal{M}_i)}{\partial \theta_t} - \bar{g}. \tag{5}$$

We further construct the task-specific matrix $\hat{G} = [\hat{g}_1, \hat{g}_2, \cdots, \hat{g}_{t-1}] \in \mathbb{R}^{|\theta_t|\times(t-1)}$, where $|\cdot|$ represents cardinality. We denote the column space of $\hat{G}$ by $\hat{\mathcal{G}}$. Since the shared component is subtracted, the dimension of $\hat{\mathcal{G}}$ is at most $t-2$. Here we assume that the number of tasks is less than the number of parameters of the model, i.e., $N < |\theta_t|$, which generally holds for deep network. The schematic illustration of gradient decomposition is presented in Figure 1(a).

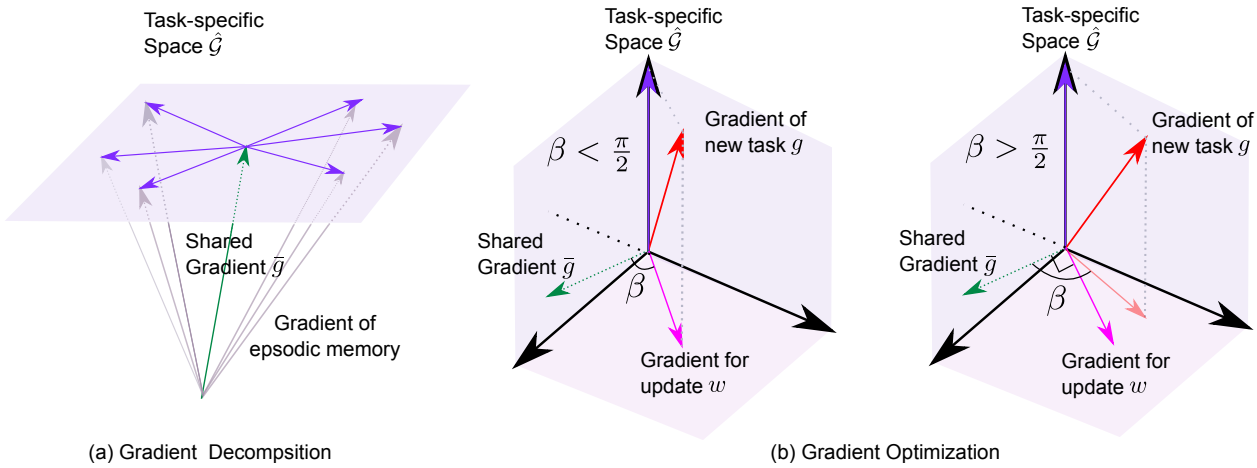
3.2. Gradient Optimization

Our algorithm aims to find a gradient w that reduces the loss of the new task but does not increase the losses of any memory tasks. Inspired by GEM, if we assume the shared loss function is locally linear, the change of loss can be diagnosed by the sign of the inner product between its corresponding gradient $\bar{g}$ and the update w. The positive, zero, or negative inner product of w and $\bar{g}$ indicates that the shared loss will decrease, preserve or increase respectively if we update the network by $-\eta w$, where η is a small positive value. Similar consideration applies to the task-specific residuals by replacing $\bar{g}$ with $\hat{g}_i$.

Based on the above observations, our algorithm requires that the gradient update w should be close to g by minimizing $\|w - g\|_2^2$ and will not increase the shared loss by constraining $\bar{g}^\top w \geq 0$, That is:

$$\min_w \quad \frac{1}{2}||w - g||_2^2, \qquad \text{s.t.} \quad \bar{g}^\top w \geq 0. \tag{6}$$

However, when considering the task-specific gradients $\hat{g}_i$, we find that the only solution to $\hat{g}_i^\top w \geq 0, \forall i < t$ is $\hat{g}_i^\top w = 0, \forall i < t$. According to Equ. (5), for any gradient update w, we have $\sum_{i=1}^{t-1} \hat{g}_i^\top w = 0$ because $\sum_{i=1}^{t-1} \hat{g}_i = \mathbf{0}$. It means that unless $\hat{g}_i^\top w = 0, \forall i < t$, any other w would result in $\hat{g}_i^\top w > 0$ for some i while $\hat{g}_i^\top w < 0$ for some other i, i.e., the losses of some tasks will increase but the losses of others will decrease. To better preserve the knowledge from each old task, the only choice is to require



(a) Gradient Decompsition (b) Gradient Optimization

Figure 1: (a) Illustration of gradient decomposition. Light grey arrow: gradients of episodic memory. Green arrow: shared gradient $\bar{g}$. Purple arrow: task-specific gradients. Every gradient of episodic memory equals to the sum of the shared gradient $\bar{g}$ and its specific gradient $\hat{g}$. Task-specific Space $\hat{\mathcal{G}}$ is spanned by all task-specific gradients. (b) Illustration of gradient optimization. β is the angle between the shared gradient $\bar{g}$ and the projection of g onto the null space of $\hat{\mathcal{G}}$. $\beta < \frac{\pi}{2}$ and $\beta > \frac{\pi}{2}$ means condition $\bar{g}^\top Pg \geq 0$ and $\bar{g}^\top Pg < 0$ respectively. If $\beta < \frac{\pi}{2}$, the gradient for update w is the projection of g onto the null space of $\hat{\mathcal{G}}$. If $\beta > \frac{\pi}{2}$, the gradient for update w should be orthogonal to both task-specific space $\hat{\mathcal{G}}$ and the shared gradient $\bar{g}$.

the gradient update w orthogonal to the column space of the task-specific gradients, i.e.,

$$\hat{G}^\top w = \mathbf{0} \tag{7}$$

For the ease of optimization, $\hat{G}^\top w = \mathbf{0}$ can be transformed according to the following lemma:

Lemma 1. *For a matrix $X \in \mathbb{R}^{N\times n}$, where rank$(X) = r$ and $r \leq n < N$, there must exist $Y \in \mathbb{R}^{N\times r}$ and $A \in \mathbb{R}^{r\times n}$ that satisfy:*

$$X = YA, \tag{8}$$

where $Y^\top Y = \mathbf{I}$, rank$(Y) = r$ and rank$(A) = r$. Moreover, for a vector $v \in \mathbb{R}^N$, $X^\top v = \mathbf{0}$ if and only if $Y^\top v = \mathbf{0}$.

Proof. See supplementary material. □

This lemma states that each column vector of an arbitrary matrix X with rank r can be represented by the linear combination of r linearly independent column vectors. Therefore, the equality constraints $\hat{G}^\top w = \mathbf{0}$ can be transferred into

$$B^\top w = \mathbf{0}, \tag{9}$$

where $B = [b_1, \cdots, b_r] \in \mathbb{R}^{|\theta_t|\times r}$ is an orthogonal matrix that can be obtained by Gram-Schmidt process. The column space of B equals to that of $\hat{G}$, i.e., $\hat{\mathcal{G}}$.

Based on the above analysis, the gradient update w can be obtained by solving the following optimization problem:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}||w - g||_2^2, \\ \text{s.t.} \quad & \bar{g}^\top w \geq 0, \\ & B^\top w = \mathbf{0} \end{aligned} \tag{10}$$

After transformation, we can solve the optimization problem in Equ. (10) by Karush-Kuhn-Tucker condition [4]. The solution is as follows:

$$w = \begin{cases} Pg, & \bar{g}^\top Pg \geq 0 \\ Pg - \frac{\bar{g}^\top Pg}{\bar{g}^\top P\bar{g}} P\bar{g}, & \bar{g}^\top Pg < 0 \end{cases} \tag{11}$$

where $P = \mathbf{I} - BB^\top$, which is positive semidefinite. The schematic representation of gradient optimization is present in Figure 1(b).

PCA relaxation: The equality constrains in the optimization problem in Equ. (10), i.e., $B^\top w = \mathbf{0}$, enforce w to be perpendicular to the whole residue space of old tasks. These constrains will become stronger as the number of old tasks increases, which may prevent w from learning the new task. To address this problem, we propose a method to relax these constrains by replacing B with its first few principal components, which can be obtained by principal component analysis. Although the above analysis defines the loss on the whole dataset, this analysis also applies to the loss on the data within each mini-batch, which replaces the whole data in Equ. (1) by the data from a mini-batch. In each mini-batch, we update the parameters by $\theta_t \leftarrow \theta_t - \eta w$, where η is the learning rate and w is obtained from Equ. (11).

3.3. Layerwise Gradient Update

Traditional gradient-based CL algorithms, e.g. GEM, concatenate gradients from all layers together and the optimization is based on this concatenated gradient. The loss decrease of episodic memory $\mathcal{M}_i$ after updating parameters θ_t by $-\eta w$ is

$$\Delta\mathcal{L}_1^{\text{old}}(\theta_t, \mathcal{M}_i) \approx -\eta \sum_l \frac{\partial \mathcal{L}^{\text{old}}(\theta_t, \mathcal{M}_i)}{\partial \theta_t} w,$$

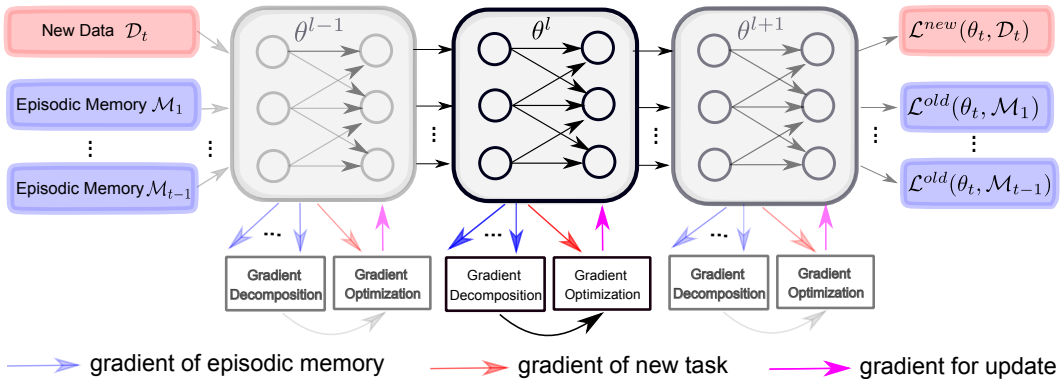


Figure 2: Illustration of layerwise gradient update strategy, which applies the proposed gradient decomposition and optimization to each layer independently.

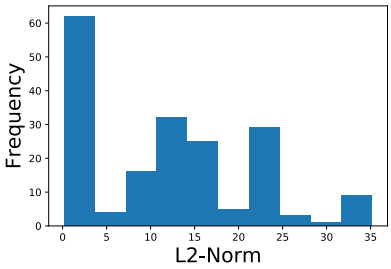


Figure 3: Histogram of the magnitude of gradients from different layers. We use L2-Norm to quantify the magnitude.

where w is obtained by Equ. (11). Considering $\frac{\partial \mathcal{L}^{\text{old}}(\theta_t, \mathcal{M}_i)}{\partial \theta_t}$ is the summation of the shared gradient $\bar{g}$ and its task-specific gradient $\hat{g}$, the loss change of episodic memory $\mathcal{M}_i$ will be negative if $\bar{g}Pg \geq 0$ holds but will be zero if $\bar{g}Pg < 0$ after replacing w with Equ (11). Specifically,

$$\Delta\mathcal{L}_1^{\text{old}}(\theta_t, \mathcal{M}_i) = \begin{cases} -\eta\bar{g}^\top Pg, & \bar{g}^\top Pg \geq 0 \\ 0, & \bar{g}^\top Pg < 0 \end{cases} \tag{12}$$

We notice that the condition $\bar{g}^\top Pg$ heavily depends on the elements with large magnitude in concatenated gradients as inner product is used. Since we have observed that the magnitude of gradients have a large variation among different layers from Figure 3, $\Delta\mathcal{L}_1^{\text{old}}(\theta_t, \mathcal{M}_i)$ will be dominated by some layers where the magnitude of new gradient g, shared gradient $\bar{g}$ and task-specific gradients $\hat{G}$ is large.

For our proposed layerwise gradient update strategy, we replace concatenated g, P in Equ.(11) with layer-specific gradients $g^{(l)}, P^{(l)}$, where l is layer index. Suppose the network is updated layerise, i.e., $\theta_t^{(l)} \leftarrow \theta_t^{(l)} - \eta w^{(l)}$, the loss change of episodic memory $\mathcal{M}_I$ is $-\eta\sum_{l=1}^{L}\bar{g}^{(l)\top}P^{(l)\top}g^{(l)\top}$, where L is the number of layers. For layer l that satisfies $\bar{g}^{(l)\top}P^{(l)\top}g^{(l)} \geq 0$, its contribution to $\Delta\mathcal{L}_2^{\text{old}}$ is $-\eta\bar{g}^{(l)\top}P^{(l)\top}g^{(l)} \leq 0$. For layer l that satisfies $\bar{g}^{(l)\top}P^{(l)\top}g^{(l)} < 0$, its contribution to $\Delta\mathcal{L}_2^{\text{old}}$ is zero. Therefore, the total loss change of episodic memory $\mathcal{M}_i$, $\Delta\mathcal{L}_2^{\text{old}}$, can be presented as

$$\Delta\mathcal{L}_2^{\text{old}} = -\eta\sum_{l\in W^+}\bar{g}^{(l)\top}P^{(l)\top}g^{(l)} < 0 \tag{13}$$

where $\{W^+\}$ is the set of layer index whose condition $\bar{g}^{(l)\top}P^{(l)\top}g^{(l)} \geq 0$ is satisfied. We notice that since the gradient magnitude will not have a large variation within one layer, $\Delta\mathcal{L}_2^{\text{old}}$ calculated layerwise is not dominated by a few elements with large magnitudes in $g^{(l)}$ and $P^{(l)}$.

Comparing Equ. (12) with Equ. (13), the layerwise method has two advantages: first, instead of heavily depending on the gradients with large magnitude in the concatenating method, the layerwise method treats each layer equally by letting each layer determine its own condition.

Second, as each layer determines its own condition, $\bar{g}^{(l)\top}P^{(l)\top}g^{(l)} \geq 0$ holds for some layers even though $\bar{g}^\top P^\top g < 0$, which may reduce the loss more efficiently.

3.4. Comparisons with GEM and A-GEM

Since GEM and A-GEM are two closest algorithms to our method, we would make more explicit comparisons with them to clarify our innovations.

Similarity. All three methods tackle "*catastrophic forgetting*" by modifying the gradients of new task under the constraints related to the gradients of old tasks.

Difference. They are different in three aspects. (1) They differ by the constraints: GEM considers the old tasks to be independent by enforcing independent losses for old tasks. A-GEM only uses a global loss for all old tasks. As admitted in [7], due to the use of single global loss, A-GEM may lead to the performance drop of some tasks. Our task-specific constraints can better exploit the inter-task dependency, as analyzed in Section 3.2. (2) They differ in the efficiency for solving the optimization problem. GEM has to solve QP online, while A-GEM and ours have closed-form solutions, which is computationally efficient. (3) Our proposed Layerwise gradient update and PCA relaxation are

not investigated in GEM or A-GEM. The advantage and motivation of our contribution on layerwise gradient update are introduced in Section 3.3.

4. Experiments

4.1. Setup

Datasets. We employ four standard benchmark datasets to evaluate the proposed continual learning framework, including MNIST Permutation, the split CIFAR10, the split CIFAR100 and the split *tiny*ImageNet datasets. The MNIST Permutation is a synthetic dataset based on MNIST [25], where all pixels of an image are permuted differently but coherently in each task. The split CIFAR10/100 dataset is generated from CIFAR10/100 [23]. The split *tiny*ImageNet is derived from *tiny*ImageNet [24]. These datasets equally divide their target classes into multiple subsets, where each subset corresponds to individual task. We considered $T = 20$ tasks for MNIST Permutation, Split CIFAR100 and Split *tiny*ImageNet, and $T = 5$ for Split CIFAR 10. For each dataset, each task contained samples from a disjoint subset of classes, except that on MNIST two consecutive tasks contained disjoint samples from the same class. The evaluation was performed on the test partition of each dataset.

Implementation Details. We trained our models following the description in GEM [30]. On the MNIST tasks, we used fully-connected neural networks with two hidden layers of 100 ReLU units. On the CIFAR10/CIFAR100 tasks, we used a smaller version of ResNet18 [15], which has three times less number of feature maps for all layers than the standard ResNet18. On the *tiny*ImageNet, we empolyed the standard ResNet18. We trained all networks using the plain stochastic gradient descent optimizer with mini-batches of 10 samples for the new task on MNIST Permutation, Split Cifar10/Cifar100 and 20 samples on *tiny*ImageNet. The batch size for each episodic memory task is 20. We followed the same one/three epoch(s) settings as GEM, where the samples in the new task were only trained once/three times but samples in the memory could be trained for several times. The learning rate was 0.1 for all datasets. The pseudo code of training is presented in Algorithm 1.

Evaluation Metrics. Following [6, 30, 7], we measure the performance by mean classification accuracy (ACC) and Backward Transfer (BWT). BWT is defined as the change of average accuracy for old tasks after learning a new task. A positive value of BWT means that learning new tasks can benefit old tasks, while a negative value indicates that learning new task degrades the performance of old tasks.

4.2. Empirical analysis

To investigate the contribution of new components in our proposed method, we incrementally evaluate each of them on MNIST Permutation, Split CIFAR10, Split CIFAR100 and Split *tiny*Imagenet. Denote **SCC** as the **S**hared **C**omponent **C**onstraint, **TSCC** as **T**ask-**S**pecific **C**ompoent **C**onstraint, **LGU** as **L**ayerwise **G**radient **U**pdate, and **PCA** as **P**rincipal **C**ompoent **A**nalysis. We choose a single predictor fine-tuned across all tasks as baseline. Seven variants are then constructed on top of baseline: (a) baseline; (b) baseline+SCC; (c) baseline+SCC+LGU; (d) baseline+SCC+TSCC; (e) baseline+SCC+TSCC+PCA; (f) baseline+SCC+LGU+TSCC; (g) baseline+SCC+LGU+TSCC+PCA. Table 1 presents the performance of all variants. All reported results are tested on the task-incremental setting.

Effectiveness of Gradient Decomposition. We compare Methods (a,b,d) in Table 1 to illustrate to what extent the shared gradient constraint and task-specific gradients constraints contribute to the performance. Split CIFAR100 and Split *tiny*Imagenet are taken as examples for analysis. We observe that the shared gradient constraint plays a significant role in continual learning task as only applying the shared gradient constraint improves the performance of the model from 54.7% to 65.3% on Split CIFAR100 and from 21.8% to 33.5% on Split *tiny*ImageNet at the final continual step. The task-specific constraints can bring additional gain for the model. It improves the mean accuracy from 65.3% to 67.8% on Split CIFAR100 and from 33.5% to 36.4% on

Algorithm 1 Training Procedures of Step t

Require:
$\{\mathcal{M}_i\}_i^{t-1}$: episodic memory of old tasks
$\mathcal{D}_t$: new data of current step t
f: network architecture
θ_{t-1}: network paramters trained by last step $t-1$
bs_{new}: batch size of the new task
bs_{old}: batch size of the old tasks
η: learning rate
Ensure:
$\mathcal{M}_t$: episodic memory of current step t
θ_t: network parameters trained by current step t

Initialization: $\theta_t = \theta_{t-1}$
for $epoch = 1 : epoch\_stop$ **do**
  for $iter = 1 : iter\_stop$ **do**
    Sample a batch s_{new} of size bs_{new} from $\mathcal{D}_t$
    $g \leftarrow \nabla\mathcal{L}(f(x;\theta_t), y)$ for $(x, y) \in s_{\text{new}}$
    for $i = 1 : t-1$ **do**
      Sample a batch s_i of size bs_{old} from $\mathcal{M}_i$
      $g^i \leftarrow \nabla\mathcal{L}(f(x;\theta_t), y)$ for $(x, y) \in s_i$
    end for
    Compute shared component $\bar{g}$ by Equ. (4)
    Compute specific components $\hat{G}$ by Equ. (5)
    if PCA Relaxation **then**
      Compute the principal components B of $\hat{G}$ by PCA
    else
      Compute the orthogonal basis B of $\hat{G}$ by Schmidt orthogonalization
    end if
    Compute the gradient for update w by Equ. (11)
    Update θ_t: $\theta_t \leftarrow \theta_t - \eta w$
  end for
end for
$\mathcal{M}_t \leftarrow$ a subset of $\mathcal{D}_t$

Method	(a)	(b)	(c)	(d)	(e)	(f)	(g)
SCC		✓	✓	✓	✓	✓	✓
TSCC				✓	✓	✓	✓
LGU			✓			✓	✓
PCA					✓		✓
MNIST	57.4	76.4	82.0	81.9	81.9	82.9	82.9
CIFAR10	58.7	75.8	76.2	76.8	77.5	77.3	79.0
CIFAR100	54.7	65.3	67.3	67.8	68.2	69.1	69.3
*tiny*Imagenet	21.8	33.5	36.3	36.4	38.1	37.6	38.3

Table 1: Empirical analysis of the proposed components on MNIST Permutation, Split CIFAR10, Split CIFAR100 and Split *tiny*Imagenet datasets in one epoch setting. (a-f) denotes different methods.

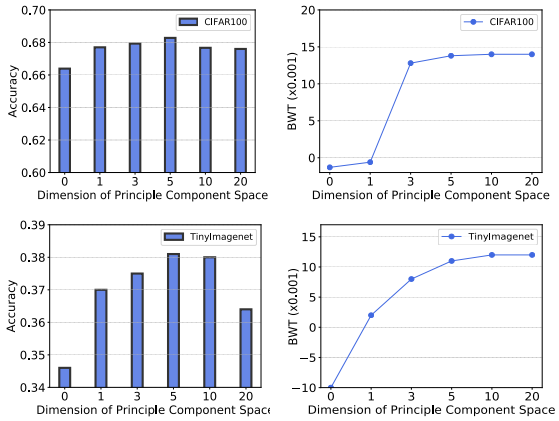


Figure 4: Average Accuracy and Backward Transfer with different dimension of principal component space. Top row: ACC and BWT on CIFAR 100. Bottom row: ACC and BWT on *tiny*ImageNet.

Split *tiny*ImageNet at the final continual step. The more enhancement of performance on both two datasets verifies that the shared gradient constraint contains most of the knowledge for old tasks and it is the most important constraint for continual learning.

Effectiveness of PCA Relaxation. To show PCA relaxation can empower our method a better trade-off between learning and memorizing, we evaluate BWT and ACC under different value of K on both Split CIFAR100 and Split *tiny*ImageNet datasets. Here we denote K as the rank of task-specific matrix after relaxation. Figure 4 shows that as K increases, ACC increases initially but decreases afterwards, while BWT keeps increasing. The continued increase of BWT is expected, as more knowledge from the old tasks is preserved as K increases. However, ACC decreases after $K = 5$, indicating that although the performance on the old tasks benefits from a large K, the ability to learn new task would be degraded if K is too large.

Effectiveness of Layer-wise Gradient Update. We propose the layer-wise gradient update strategy where the constraint is imposed by individual network layer. We compare

Datasets	LGU	GEM		A-GEM		Method (d)	
		ACC	BWT	ACC	BWT	ACC	BWT
MNIST	x	81.9	0.017	76.4	0.005	81.9	0.017
	✓	**82.7**	**0.045**	**77.1**	**-0.008**	**82.9**	**0.045**
CIFAR10	x	76.8	0.014	75.8	0.007	76.8	0.015
	✓	**77.4**	**0.035**	**76.5**	**0.015**	**77.3**	**0.034**
CIFAR100	x	65.8	-0.005	65.3	-0.005	67.8	-0.001
	✓	**67.9**	**0.003**	**66.8**	**0.002**	**69.1**	**0.006**
*tiny*Imagenet	x	34.2	-0.001	33.5	-0.003	36.4	0.008
	✓	**35.8**	**0.006**	**35.6**	**0.005**	**37.6**	**0.018**

Table 2: Empirical Analysis of layerwise gradient update on GEM, A-GEM and Method (d).

Methods (b,c) in Table 1 to manifest its general effectiveness. The results show that with layer-wise gradient update strategy, Average Accuracy ACC is higher than those by the method without layer-wise gradient update strategy. In addition, from method (b-e) in Table 1, we can conclude that Gradient decomposition (SCC, TSCC) and Layerwise Gradient Update (LGU) are orthogonal and cumulative. To further illustrate the effectiveness of layerwise gradient update strategy, we leverage layerwise gradient update on top of GEM, A-GEM and our proposed method (d) and present the results in Table 2. By comparing GEM, A-GEM, Method (d) with their layerwise gradient update counterparts, we observe that both ACC and Backward Transfer (BWT) of algorithms with layerwise gradient update are systematically higher than those without the strategy. The results empirically prove the general effectiveness of layerwise gradient update on GEM Families.

4.3. Main results

Our main results except MNIST are tested on the online task-incremental setting while MNIST is tested on the data-incremental setting. We compare our proposed method with several existing and state-of-the-art methods, including Single [40], Independent[1] [30], Multimodel [30], EWC [22], iCARL [38], GEM [30], A-GEM [7] and S-GEM [7]. For fair comparison, the size of each episodic memory is also 256 for GEM, A-GEM and S-GEM. The results are reported in Tab. 3. Our method achieves state-of-the-art results on all four datasets in both one epoch and three epochs settings.

MNIST Permutation. Our method achieves 82.9% for one epoch setting and 84.3% for three epochs setting, which outperforms the competitive GEM [30] by 1.0% and 0.2%, respectively. We only marginally outperforms GEM at the three epochs setting because the network has been saturated.

Split CIFAR10. On Split CIFAR10, our method significantly outperforms GEM by 2.2% in one epoch setting. In three epochs setting, our method achieves similar results with S-GEM where we only improve by 0.1% on Split

[1]For fair comparison, we follow the setting in GEM. In GEM, the channels of each feature map are divided by the number of tasks. As such, the model size in Independent method equals to the model size in other methods.

Method	MNIST Permutation		Split CIFAR10		Split CIFAR100		Split *tiny*ImageNet	
	1 epoch	3 epochs	1 epoch	3 epochs	1 epoch	3 epochs	1 epoch	3 epochs
Single [40]	57.4 ± 0.7	53.0 ± 0.6	57.7 ± 0.3	68.4 ± 0.8	54.7 ± 1.0	55.4 ± 0.9	25.7 ± 0.8	28.7 ± 0.5
Independent [30]	37.0 ± 0.6	61.3 ± 0.4	43.3 ± 0.8	71.9 ± 0.7	43.3 ± 0.3	53.5 ± 0.5	26.9 ± 0.6	35.9 ± 0.6
Multimodel [30]	71.9 ± 0.5	59.4 ± 0.5	-	-	-	-	-	-
EWC [22]	57.5 ± 0.6	61.9 ± 0.5	59.8 ± 0.4	67.8 ± 0.5	48.3 ± 0.6	56.9 ± 0.6	22.2 ± 0.4	25.4 ± 0.3
iCARL [38]	-	-	63.9 ± 0.5	66.5 ± 0.6	55.7 ± 0.8	55.8 ± 0.4	26.5 ± 0.6	28.9 ± 0.4
GEM [30]	81.9 ± 0.4	84.1 ± 0.4	76.8 ± 0.3	80.8 ± 0.3	65.8 ± 0.5	69.6 ± 0.4	34.2 ± 0.5	37.3 ± 0.4
A-GEM [7]	76.4 ± 0.3	82.9 ± 0.5	75.8 ± 0.6	81.0 ± 0.4	65.3 ± 0.4	69.1 ± 0.5	33.5 ± 0.3	36.9 ± 0.3
S-GEM [7]	76.3 ± 0.4	82.3 ± 0.4	75.9 ± 0.3	81.5 ± 0.3	64.2 ± 0.3	68.8 ± 0.2	33.8 ± 0.4	36.1 ± 0.3
tinyER† [8]	82.5 ± 0.8	-	77.5 ± 0.5	-	68.3 ± 0.7	-	36.5 ± 0.3	-
GDumb† [37]	73.4 ± 0.4	-	74.2 ± 0.3	-	60.3 ± 0.9	-	32.1 ± 0.3	-
Ours	$\mathbf{82.9 \pm 0.3}$	$\mathbf{84.3 \pm 0.3}$	$\mathbf{79.0 \pm 0.4}$	$\mathbf{81.6 \pm 0.5}$	$\mathbf{69.3 \pm 0.4}$	$\mathbf{71.0 \pm 0.3}$	$\mathbf{38.3 \pm 0.4}$	$\mathbf{39.5 \pm 0.3}$

Table 3: Comparison of our proposed method with state-of-the-art methods on MNIST Permutation, Split CIFAR10, Split CIFAR100 and Split *tiny*ImageNet in Online Task Incremental setting. All the experiments are conducted in three runs. †indicates that we follow the training schedule of Online Task Incremental Disjoint setting in [8] and [37] respectively, which means every new sample can only be trained once.

CIFAR10, which is within statistical error. The marginal improvement can be explained by that the Split Cifar10 is oversimple and various medels can touch the upper bound accuracy of each task.

Split CIFAR100. On Split CIFAR100, we improve the ACC from 65.8% to 69.3% by 3.5% in one epoch setting. In three epochs setting, we improve ACC from 69.6% to 71.0% by 1.4%.

Split *tiny*ImageNet. As illustrated in Tab. 3, we significantly improve the benchmark by 3.9%, which is from 34.2% to 38.1% in one epoch setting. In three epoches setting, our method also outperforms GEM by 2.6%.

4.4. Results for the class-incremental setting

Different from task-incremental continual learning setting mainly discussed in this paper, class-incremental continual learning setting do not provide task identity in the test data [19, 52]. In such setting, task boundaries naturally formed in the continual training set ought to be broken because we need classify classes from different continual tasks. In order to break the task boundary, we first stack all episodic memories as one replay buffer. When training, we randomly split the replay buffer into N sub-replay buffers. The gradient decomposition is implemented on such N sub-replay buffers instead of episodic memories. We assess our method on Split CIFAR100 dataset for consistency with other approaches. We employ Resnet34 as our backbone and train the network for 200 epochs for each continual step. The initial learning rate is 0.1 and decays when the epoch equals 80, 160. The size of replay buffer is restricted to be 2000 images and the memory update strategy is the same as that in [38, 5] for fair comparison.

We compare our approach with other competing methods, including LWF [29], iCarl [38], DR [17], End2End [5], LUCR[2] [18], GD [26], MUC [11] and TPL [51]. We fix $N = 5$ when implementing the experiments. The comparisons with other competing methods are presented in Table 4. As illustrated in Table 4, our method outperforms GEM by 3% which shows the consistency improvement with results in task-incremental scenario. We also achieve the state-of-the-art results which are the same MUC [11].

[2]LUCR is short for Learning a Unified Classifier via Rebalancing

Method	Ref	Results	Method	Ref	Results
LwF[a]	PAMI' 17	58.4	GD	ICCV' 19	62.1
iCarl	CVPR' 17	58.7	A-GEM[b]	ICLR' 19	60.43
DR[a]	ECCV' 18	59.1	S-GEM[b]	ICLR' 19	59.98
End2End[a]	ECCV' 18	60.2	GEM[b]	NIPS' 17	61.9
LUCR	CVPR' 19	61.2	TPL	ECCV'20	**65.3**
MUC	ECCV' 20	64.7	Ours	-	**65.3**

[a] LwF and DR were adapted for class-incremental setting. Results of LwF, DR and End2End were reported in [26].
[b] GEM, A-GEM, S-GEM are re-implemented by ourselves.

Table 4: Comparison of our proposed method with state-of-the-art methods on Split CIFAR100 in the class-incremental setting.

5. Conclusion

In this work, we presented a novel continual learning framework including gradient decomposition, gradient optimization and layerwise gradient update. The gradients of episodic memory are decomposed into the shared gradient and task-specific gradients. Our framework encourages the consistency between the gradient for update and the shared gradient, and enforces the gradient for update orthogonal to the space spanned by task-specific gradients. The former keeps the common knowledge, while the latter can be relaxed by PCA to preserve the task-specific knowledge. We observe that optimizing the gradient update layerwise can further help the model remember the old tasks. Our method significantly outperforms current state-of-the-art on extensive benchmarks.

6. Acknowledgement

Wanli Ouyang was supported by the SenseTime, Australian Research Council Grant DP200103223, and Australian Medical Research Future Fund MRFAI000085.

References

[1] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018. 2

[2] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10(Jul):1737–1754, 2009. 2

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 2

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 4

[5] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018. 1, 2, 8

[6] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 6

[7] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018. 1, 2, 5, 6, 7, 8

[8] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. 8

[9] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5138–5146, 2019. 2

[10] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. 2

[11] E. Fini, S. Lathuilière, E. Sangineto, M. Nabi, and E. Ricci. Online continual learning under extreme memory constraints. In *European Conference on Computer Vision*, pages 720–735. Springer, 2020. 2, 8

[12] S. Flennerhag, A. A. Rusu, R. Pascanu, H. Yin, and R. Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019. 2

[13] Y. Ge, D. Chen, and H. Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. In *International Conference on Learning Representations*, 2020. 2

[14] Y. Ge, F. Zhu, D. Chen, R. Zhao, and H. Li. Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. In *Advances in Neural Information Processing Systems*, 2020. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[16] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[17] S. Hou, X. Pan, C. Change Loy, Z. Wang, and D. Lin. Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 437–452, 2018. 2, 8

[18] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019. 2, 8

[19] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018. 8

[20] H. Jung, J. Ju, M. Jung, and J. Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016. 2

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 2, 7, 8

[23] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[24] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015. 6

[25] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 6

[26] K. Lee, K. Lee, J. Shin, and H. Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 312–321, 2019. 2, 8

[27] Y. Lee and S. Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2933–2942, 2018. 2

[28] S. Li, D. Chen, B. Liu, N. Yu, and R. Zhao. Memory-based neighbourhood embedding for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[29] Z. Li and D. Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2, 8

[30] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017. 1, 2, 6, 7, 8

[31] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[32] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 2

[33] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017. 2

[34] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019. 1

[35] D. Park, S. Hong, B. Han, and K. M. Lee. Continual learning by asymmetric loss approximation with single-side overestimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3335–3344, 2019. 2

[36] A. Pentina and C. Lampert. A pac-bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pages 991–999, 2014. 2

[37] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, pages 524–540. Springer, 2020. 8

[38] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1, 2, 7, 8

[39] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[40] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. 7, 8

[41] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 2

[42] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[43] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017. 1

[44] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3400–3409, 2017. 1

[45] D. L. Silver, Q. Yang, and L. Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013. 2

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[47] S. Sodhani, S. Chandar, and Y. Bengio. On training recurrent neural networks for lifelong learning. *arXiv preprint arXiv:1811.07017*, 2018. 2

[48] P. Su, K. Wang, X. Zeng, S. Tang, D. Chen, D. Qiu, and X. Wang. Adapting object detectors with conditional domain normalization. 2020. 2

[49] S. Tang, D. Chen, L. Bai, Y. Ge, and W. Ouyang. Mutual crf-gnn for few shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[50] S. Tang, P. Su, D. Chen, and W. Ouyang. Gradient regularized contrastive learning for continual domain adaptation. *arXiv preprint arXiv:2103.12294*, 2021. 1

[51] X. Tao, X. Chang, X. Hong, X. Wei, and Y. Gong. Topology-preserving class-incremental learning. Springer, 2020. 8

[52] G. M. van de Ven and A. S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 8

[53] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 2

[54] Y. You, I. Gitman, and B. Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 2

[55] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017. 2

[56] C. Zeno, I. Golan, E. Hoffer, and D. Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018. 2

[57] B. Zhao, S. Tang, D. Chen, H. Bilen, and R. Zhao. Continual representation learning for biometric identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1198–1208, 2021. 1

[58] J. Zhu, S. Tang, D. Chen, S. Yu, Y. Liu, et al. Complementary relation contrastive distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2