Measuring Asymmetric Gradient Discrepancy in Parallel Continual Learning

Fan Lyu, Qing Sun, Fanhua Shang, Liang Wan, Wei Feng*
College of Intelligence and Computing, Tianjin University
{fanlyu, sssunqing, fhshang, lwan}@tju.edu.cn, wfeng@ieee.org

Abstract

In Parallel Continual Learning (PCL), the parallel multiple tasks start and end training unpredictably, thus suffering from both training conflict and catastrophic forgetting issues. The two issues are raised because the gradients from parallel tasks differ in directions and magnitudes. Thus, in this paper, we formulate the PCL into a minimum distance optimization problem among gradients and propose an explicit Asymmetric Gradient Distance (AGD) to evaluate the gradient discrepancy in PCL. AGD considers both gradient magnitude ratios and directions, and has a tolerance when updating with a small gradient of inverse direction, which reduces the imbalanced influence of gradients on parallel task training. Moreover, we present a novel Maximum Discrepancy Optimization (MaxDO) strategy to minimize the maximum discrepancy among multiple gradients. Solving by MaxDO with AGD, parallel training reduces the influence of the training conflict and suppresses the catastrophic forgetting of finished tasks. Extensive experiments validate the effectiveness of our approach on three image recognition datasets in task-incremental and class-incremental PCL. Our code is available at https://github.com/fanlyu/maxdo.

1. Introduction

Continual Learning (CL) [25, 27, 31, 43], aims to continuously learn new knowledge from a sequence of tasks with non-overlapping data streams over a lifelong time. In the era of Internet of Things, people are using many smart devices, where multi-source data and tasks would be accessed at any time. A CL system should respond to parallel data streams from multiple devices. We study Parallel Continual Learning (PCL), as shown in Fig. 1, where an unfixed number of tasks are trained in a parallel way at any time. Specifically, according to the access time of each task, PCL builds an adaptive number of parallel data pipes, thus enabling instant response to new-coming tasks without pending.

Due to the parallel data streams from different tasks, PCL suffers from not only the catastrophic forgetting but the training conflict among parallel tasks. Most existing methods in CL are proposed to tackle the catastrophic forgetting [19, 25], including regularization-based [25, 8, 16, 51, 1], rehearsal-based [31, 9, 21, 4, 41, 36], and architecture-based [33, 49, 39, 38] methods. In PCL, the training processes of different tasks are diverse, i.e., each task starts and ends training unpredictably (see Fig. 1). Thereby the gradient from different task differs in direction and magnitude [50] and may be neutralized. The gradient discrepancies lead to catastrophic forgetting and training conflict issues, which may fail the learning of some tasks. At any time in PCL, therefore, we present that the problem can be formulated to find an optimal gradient in a minimum distance multi-objective optimization, where each objective is to minimize the distance to a target gradient. In general, the distance metric is proportional to the effect of the optimal gradient on the corresponding task.

In most situations, the mentioned distance metric \( D \) between gradients is set to symmetric intuitively, such as the Euclidean distance and cosine distance. In other words, we usually have \( D(x, y) = D(y, x) \) for any \( x \) and \( y \). However, the gradient influence is imbalanced among parallel tasks in the gradient descent. For example, in Fig. 1, at the marked time, we have three gradients with diverse directions and magnitudes, and updating with any of them provides different influences to the other two. In the minimum distance problem, the optimal solution should have the minimum negative influence on all parallel tasks, but using symmetric metrics means the influences are optimized indistinguishably at the same time. Due to the fact that the gradients are with wide differences, the solution may have large biases, which would get the near-fitting task out of its local minimum but has less impact on a new-coming task.

To measure the gradient discrepancy, we hold the opinion that the distance metric in the min-distance problem should be asymmetric. First, though the metric is bound up with both the gradient magnitude and direction, the influences on model training from gradients should be asymmetric, where the model should have more tolerance to small gradients even if they indicate an inverse direction. Second, because
Data Stream A for task A
Data Stream B for task B
Data Stream C for task C
Gradient Discrepancy
Parallel Continual Learning
Asymmetric distance
Minimize Asymmetric Distance
Gradient Discrepancy
Optimal gradient

Our main contributions are three-fold:

1. We formulate the PCL into a minimum distance problem and compare symmetric and asymmetric distances. We show that symmetric metrics are not effective in solving the problem and suggest asymmetric metrics.

2. We propose an asymmetric metric, named AGD, to evaluate the gradient discrepancy, which is proportional to the gradient magnitude ratios and directions. AGD measures the imbalance of gradient influence in PCL.

3. We propose MaxDO for minimizing gradient discrepancy of different tasks, which maximally reduces the asymmetric discrepancy from a gradient to the others. MaxDO avoids the self-interference among gradients and reduces training conflict and catastrophic forgetting.

2. Related Work

2.1. Continual Learning

Continual Learning (CL) represents receiving data from new domains continually. In traditional CL, the new domains show up one by one, say serial CL. CL methods can be classified into three kinds. (1) Rehearsal [31, 9, 21, 4, 41, 36, 32], which saves or generates data of old tasks for retraining together with the current training. (2) Regularization [25, 8, 16, 51, 1, 17], which leverages extra regularization terms to consolidate previous knowledge when learning new tasks. (3) Dynamic architecture [33, 49, 39, 38], which freezes task-specific parameters and grows new branches for new tasks automatically. However, most of the existing CL methods are designed for reducing catastrophic forgetting in the serial scenario. Contrastively, in PCL, we need to tackle not only catastrophic forgetting but training conflict among parallel tasks, which is somehow related to multi-task learning.

2.2. Multi-Task Learning

Multi-Task Learning (MTL) [7] is used to address multiple tasks with a single model from one to many domains. Traditional MTL solutions can be mainly grouped into feature-based and parameter-based approaches [52]. The feature-based approaches focus on learning common feature representations for multiple tasks [34, 45]. The parameter-based approaches use model parameters in a task to help learn model parameters in other tasks [44, 5, 23]. Recently, some MTL methods propose to find an optimal gradient for updating and can be categorized into three types. (1) Learning-based methods [11], which learn a set of weights by back-propagation. (2) Solving-based methods [40, 29], solve the problem by finding an optimal gradient that is not dominated by the gradient from any task. (3) Calculating-based methods [30, 24, 12, 46, 50, 20, 28] compute the gradient weights
by combining gradients or losses of all tasks. Inspired by MTL, we also formulate the problem into finding an optimal gradient. Specifically, we consider that the optimal gradient should have a small distance to all gradients.

2.3. Asymmetric Metric

In most situations in neural network, the distance is set to symmetric, e.g., the Euclidean distance. However, the symmetric metric is not always suitable for finding the optimal gradient (see the section for details). Asymmetric metric [14, 35], also known as quasi-metric [14] or pseudo metric [18, 6], is a generalization of a metric but the symmetry axiom is eliminated in the definition of metric spaces. A classical example of asymmetric metric is the taxicab geometry topology including one-way streets, where a path from point A to B has different streets compared to a path from B to A. In this paper, we propose to measure the gradient discrepancy using an asymmetric metric and raise a novel optimization strategy to minimize the maximum discrepancy.

3. Our Approach

3.1. Parallel Continual Learning

On a timeline, given a sequence of \( T \) tasks with parallel data streams \( \{D_1, \ldots, D_T\} \) for continual training, and each data stream can be accessed and suspended at any time. For simplest, we assume each data stream is i.i.d., and tasks are accessed in order from 1 to \( T \) and there exists no real gap that no data stream flows on the timeline. Note that traditional CL is an edge situation of PCL that all tasks are nose-to-tail. A PCL model contains a shared backbone with parameter \( \theta \) to learn task-agnostic knowledge and adaptively incremental number of task-specific classifiers with parameters \( \theta_i \). When a new task is accessed, a corresponding task-specific classifier will be constructed.

In PCL, a task will be forgotten by learning any other tasks when its data stream ends. To avoid forgetting, we leverage the popular rehearsal strategy [31, 9, 21, 4, 41, 36] in our training. Rehearsal builds an extra data stream sampled from all seen tasks and retrains them to suppress the forgetting of finished tasks. For convenience, we denote the rehearsal data stream as \( D_0 \). At time \( t \), we use \( T_t \) to represent the activated data streams (including \( D_0 \)). Together with the rehearsal data stream, PCL training yields the following dynamic multi-objective empirical risk minimization:

\[
\min_{\theta, \{	heta_i\}_{i \in T_t}} \{ \ell_i(D_i) | i \in T_t \}.
\]

Because the task-specific classifiers are updated by their own gradients \( \theta_i \leftarrow \theta_i - \alpha_i \nabla_{\theta_i} \ell_i | i \in T_t \) with step size \( \alpha_i \), we focus on the update of the shared backbone \( \theta \). At any PCL step, the goal of dynamic MOO is to optimize multiple objectives simultaneously while updating only once, and the only update of the shared parameters depends on the gradients of all in-training tasks. It will exit an uncertain number of tasks, and each task will provide a task-specific gradient on the shared parameter \( \theta \). Let \( g_i = \nabla_\theta \ell_i \) and \( \alpha \) be a step size for optimization. The problem of the backbone update can be formulated as follows:

\[
\theta \leftarrow \theta - \alpha d^*, \quad \text{where } d^* = f(\{g_i | i \in T_t\}).
\]

The key question is how to compute the optimal gradient \( d^* \) via the function \( f(\cdot) \). In this paper, we define the function \( f(\cdot) \) as a min-distance multi-objective problem by minimizing the gradient distance from all in-training tasks:

\[
d^* = \arg \min_{d} \{ D(d, g_i) | i \in T \},
\]

where we need to identify what distance metric \( D \) is used to measure gradient discrepancy. The motivation of Eq. (3) is that for the task \( i \) in PCL, its own gradient \( g_i \) is the most qualified update direction for itself. The solution \( d^* \) should be as close to every gradient as possible.

A related but different task is Federated Continual Learning (FCL) [48, 42]. Most FCL methods primarily focus on serial training, involving multiple clients in training a shared task at the same time. It is meaningful to study Federated PCL in the future for the advancement of privacy protection.

3.2. Measuring Asymmetric Gradient Discrepancy

To measure gradient discrepancy, the Euclidean Distance (EuDist, \( D(x, y) = \|x - y\| \in [0, \infty) \)) and Cosine Distance (CosDist, \( D(x, y) = 1 - \frac{x^\top y}{\|x\|\|y\|} \in [0, 2] \)) are the two most popular choices. Both of them are symmetric, i.e., \( D(x, y) = D(y, x) \). A symmetric metric \( D(x, y) \) means the forward influence (x to y) and backward influence (y to x) are equal. For example, given two in-training tasks A and B, the distance \( D(g_A, g_B) \) represents both the effect of \( g_A \) on task B and \( g_B \) on task A because of \( D(g_A, g_B) = D(g_B, g_A) \). Note that large distance from \( g_A \) to \( g_B \) means large negative influence on the training of task B with \( g_A \).

However, the model update is highly related to gradient magnitude and direction, which are asymmetric to model updating. The influence of the gradient \( g_A \) on task B may be quite different from that of the gradient \( g_B \) on task A. In previous studies [31, 9, 50], the two tasks are treated as conflict and \( \langle g_A, g_B \rangle < 0 \). In PCL, due to the diverse training process, gradients from parallel tasks are diverse in magnitude and direction. When \( \|g_A\| \ll \|g_B\| \), the gradient \( g_A \) will have little negative influence on task B even if \( \langle g_A, g_B \rangle < 0 \); when \( \|g_A\| \gg \|g_B\| \) (e.g., a new task A is accessed when task B has been trained for some time near convergence), the update produces huge impact on task B even if \( \langle g_A, g_B \rangle > 0 \). Using traditional symmetric distances can hardly represent the asymmetric update influence difference.

To effectively measure gradient discrepancy in PCL, we introduce the asymmetric metric.
Given two gradient $g_a$ and $g_b$, the asymmetric metric such as Euclidean distance. Specifically, $g_A$ may make task B worse if $\hat{D}(g_A, g_B)$ is large (close to 1). If $\hat{D}(g_A, g_B)$ is close to 0, $g_A$ and $g_B$ has less conflict. Moreover, Corollary 1 involves that when $\|g_A\| \ll \|g_B\|$, AGD has a tolerance $\frac{1}{2}$ even if $(g_A, g_B) < 0$, which means the impact of $g_A$ on task B is mild. This is because updating with a zero gradient will neither improve nor damage the performance. Even though, we prefer positive influence rather than non-influence. Thus, we define that the distance $\hat{D}(g_A, g_B)$ in this situation is the mid-level ($\frac{1}{2}$) in the value range([0,1]). See different tolerances in our experiments.

Moreover, we compare AGD (Fig. 2(c)) with Euclidean and cosine distance in Fig. 2. First, the cosine distance (Fig. 2(a)) is magnitude irrelevant, which ignores the magnitude difference in PCL. Second, the Euclidean distance (Fig. 2(b)) depends heavily on the magnitude value difference, but ignores that the gradient influence on the model update is asymmetric. For example, when $\|x\| \to 0$, EuDist will get large if we have large $\|y\|$ without any tolerance, which ignores the non-influence of zero gradient.

### 3.3. Maximum Discrepancy Optimization

At time $t$, let the optimal solution to Problem (3) be $d^*$, where $T_i$ is the index set of in-training tasks ($T$ for simplicity). However, directly optimizing the problem is difficult due to the large decision space that has the same dimension as $\theta$. Following [28, 40], we use linear scalarization to solve the transformed problem that allows only optimizing decision variable $w \in \mathbb{R}^{|T|}$. That is, let $d = \sum_{i \in T} w_i g_i$, where $\forall w_i \geq 0$ and $\sum_{i \in T} w_i = 1$, we have

$$w^* = \arg \min_w \{ \hat{D} \left( \sum_{j} w_j g_j, g_i \right) \forall i \in T \}. \quad (4)$$

Each objective of the dual problem will be highly affected by the minimum discrepancy, i.e., each gradient itself. For example, by minimizing objective $\hat{D}(\sum_j w_j^i g_j, g_i)$, weight $w_i$ is more likely to be activated than others. Thus, multiple
As shown in Fig. 3, we propose Maximum Discrepancy Optimization (MaxDO) to reduce the maximum gradient discrepancy. Specifically, instead of the weight vector \( \mathbf{w} \in \mathbb{R}^{T \times 1} \), we optimize a weight matrix \( \mathbf{W} \in \mathbb{R}^{T \times |T|} \), in which \( \forall \mathbf{w}_{ij} \geq 0 \). \( \mathbf{W} \) can be combined by a diagonal vector \( \mathbf{w} = [\mathbf{w}_{1,1}, \cdots, \mathbf{w}_{|T|,|T|}] \) and an off-diagonal matrix \( \mathbf{W} = \mathbf{W} - \text{Diag}(\mathbf{w}) \), where \( \sum_{i \in T} \tilde{\mathbf{w}}_i = 1 \) and \( \sum_{j \in T} \tilde{\mathbf{w}}_{ij} = 1, \forall i \). Thus, \( \sum_{i,j \in T} \tilde{\mathbf{w}}_{ij} = |T| + 1 \) and the two weights are independent and can be optimized without disturbance as follows: (1) \( \mathbf{W} \), computed by Stochastic Gradient Descent (SGD), is used to make up the maximum gradient discrepancy. The objectives of any two rows in \( \mathbf{W} \) are different. For row \( i \), to formulate the maximum discrepancy of gradient \( g_i \), the objective is the combination of non-diagonal entries. The weighted other gradients should be with the smallest asymmetric distance to \( g_i \). (2) \( \mathbf{w} \) is obtained by the Multiple Gradient Descent Algorithm (MGDA) [15], which is to obtain a weighted gradient that does not damage any tasks with a min-norm optimization. The objective of MGDA is 0 and the resulting point satisfies the Karush–Kuhn–Tucker condition or the solution gives a Pareto descent direction that improves all tasks. See Appendix for more details of MGDA. For each off-diagonal entry of the \( i \)-th column, their sum means the effect of the gradient \( g_i \) reducing the maximum discrepancy from other gradients. MGDA is used to reduce the possible negative effect in MaxDO. On the other hand, MaxDO reduces the training failure of new tasks in MGDA. To sum up, our MaxDO with AGD can be computed by

\[
\mathbf{W}^* = \arg \min_{\mathbf{w}} \sum_{i \in T} \tilde{D} \left( \sum_{j \neq i} \tilde{\mathbf{w}}_{ij} g_j, g_i \right) + \text{Diag} \left( \arg \min_{\mathbf{w}} \sum_{j} \tilde{\mathbf{w}}_{ij} g_j \right). 
\]  

In Eq. (5), we can obtain an approximate solution by combining the closed-form solution and the iterative solution. Fig. 3 reveals the diagram of solving MaxDO. We project the solution of SGD onto the feasible set (\( \sum_{i,j \in T} \tilde{\mathbf{w}}_{ij} = 1 \)) via softmax at each step in the multiple steps for fast convergence. First, we initialize all entries of \( \mathbf{W} \) by \( \frac{1}{|T|} \). Then, the off-diagonal matrix is used to minimize the maximum gradient discrepancy via SGD and the diagonal vector is optimized by min-norm. Finally, the final weights are reduced to a vector by dividing \( |T| + 1 \) to guarantee that their sum.
is 1. Note that, MaxDO is implemented only when \( |T| > 1 \), i.e., multiple tasks are given at the current time. Otherwise, we have \( d^* = g_1 \) for the only current task 1. Thus, the final gradient \( d^* \) is computed by

\[
d^* = \begin{cases} 
g_1, & |T| = 1, \\
\sum_i \left( \frac{1}{|T| + 1} \sum_j W^*_{j,i} \right) g_i, & |T| > 1.
\end{cases}
\] (6)

The detailed algorithm is shown in Algorithm 1. With the rehearsal data stream, our algorithm learns a PCL model through a timeline. At the time \( t \) on the timeline, given a mini-batch \( B \) from each data stream, we compute the corresponding gradients on shared and task-specific parameters. The task-specific parameters are updated directly and the gradients on the shared backbone are collected for computed the final updated gradient \( d \). By using our MaxDO, we update the shared parameters \( \theta \) with the optimal \( d^* \).

4. Experiment

4.1. Dataset

In our experiments, 3 traditional image recognition datasets are transformed into parallel data streams: (1) Parallel Split EMNIST (PS-EMNIST). We split EMNIST [13] (62 classes) into 5 tasks and the size of the label set for each task, i.e., the number of classes, is set to \{12, 12, 12, 13, 13\}. (2) Parallel Split CIFAR-100 (PS-CIFAR-100). We split CIFAR-100 into 10 tasks and the size of the label set for each task is set to 10. (3) Parallel Split ImageNet-TINY (PS-ImageNet-TINY). We split Tiny ImageNet [26] (200 classes) into 10 tasks, and the size of the label set for each task is set to 20. We evaluate PCL on task-incremental and class-incremental scenarios. Please see Appendix for more dataset details.

All three datasets have 3 different label sets (3 different class splits), each of which has 3 different timelines (when to access). For each timeline, we have 3 different runs with fixed seeds 1234, 1235, and 1236 for parameter initialization. In other words, we have 27 different settings for each dataset, and we report the average and standard deviation (avg \( \pm \) std) for each compared method in our experiments. Note that, we omit all blank time that no data stream flows for simplicity.

4.2. Experiment Details

We implement our experiments using Tensorflow and conduct on a single NVidia RTX 3090Ti GPU card. We take a 2-layer MLP as the backbone network for PS-EMNIST and a Resnet-18 [22] for PS-CIFAR-100 and PS-ImageNet-Tiny. The learning rate is set to 0.003, 0.0004 and 0.0005 for PS-EMNIST, PS-CIFAR-100 and PS-ImageNet-Tiny. The SGD in MaxDO has a learning rate of 5. Each task is trained in a data stream, i.e., each data point passes only once. For each task, we set the batch size to 128 per step. For any new task in PCL, we build a new classifier, which is a fully-connected layer with a softmax function.

To evaluate PCL, we compute the average accuracy and forgetting, following previous continual learning studies [31, 9, 3, 2, 37]. Let \( e_t \) be the end time of task \( t \) and the final time \( \bar{e} = \max(e_1, e_2, \cdots, e_T) \), the two metrics are computed as:

\[
A_e = \frac{1}{T} \sum_{t=1}^{T} a^*_t, \quad F_e = \frac{1}{T} \sum_{t=1}^{T} a^*_{t} - a^*_{e_t},
\] (7)

where \( a^*_t \) is the mean testing accuracy of task \( j \) at time \( t \), \( A_e \) denotes the final average accuracy on all the tasks, and \( F_e \) (also known as backward transfer) means the final performance drop compared to each task that was first trained.

4.3. Main Results

We compare our method with MTL methods including MGDA [15], GradNorm [11], DWA [30], GradDrop [12], PCGrad [50] and RLW [28] in the PCL setting. We treat any time on the timeline as an MTL subunit to train PCL. All results of previous MTL methods are produced by ourselves with the claimed design in their papers. We also compare with some rehearsal-based SCL methods including AGEM[9], GMED[37] and ER[10]. To adapt to the SCL methods, we merge batches from every in-training task to mimic a naive sequence learning that only a batch from the current task and a batch from the memory buffer.

We show the main comparisons with the proposed methods in Tables 1 and 2 on the three datasets. We have several major observations. First, the rehearsal strategy is useful for reducing catastrophic forgetting in PCL for all compared methods. As an extra data stream apart from in-training data streams, rehearsal provides data from the finished tasks training together with other tasks to suppress forgetting. With the rehearsal strategy, the memory may provide continual learning of finished tasks, and even better performance can be obtained, which results in positive forgetting values on \( F_e \). Second, without task-id, class-incremental PCL has worse performance than task-incremental PCL. Surprisingly, we find that MGDA has good performance than other methods in
class-incremental PCL compared with the task-incremental scenario, which means the suppressing on catastrophic forgetting is more important in class-incremental PCL. Third, the compared MTL methods are designed for balanced training and ignore the diverse training process in PCL, thus some gradients may be counteracted because of the large gradient discrepancy when updating the model. In contrast, our MaxDO with AGD obtains the best final accuracy \( A^T \) on three datasets and two scenarios, which shows our superiority in balancing plasticity and stability. For example, we have 40.770% and 9.431% for PS-ImageNet-TINY on two scenarios respectively, while the compared best value is only 39.427% and 8.318%. On one hand, the proposed AGD is used to measure the asymmetric distance between gradients to boost the effective update of each task. On the other hand, the maximum discrepancies between multiple tasks are reduced. Note that, the forgetting measure of the proposed methods may not outperform the compared methods because we got both better new tasks and final accuracy performance, their difference value may be small.

### 4.4. Rehearsal Analysis in PCL

In Fig. 4, we compare the effects of different memory buffer sizes (per class) in rehearsal on PS-CIFAR-100. We observe that the memory buffer size affects the remembering of old knowledge, and a larger size means better knowledge keeping, which is similar to traditional CL. Under the same memory size, our method has better performance. Then, in Table 3(a), to show the MaxDO’s effectiveness of forgetting reduction on rehearsal gradient, we evaluate the result that only leverages MaxDO on new tasks. In this case, the final gradient is calculated by \( \mathbf{d} = \frac{1}{2} \mathbf{g}_{\text{re}} + \frac{1}{2} \mathbf{g}_{\text{new}} \), where \( \mathbf{g}_{\text{new}} \) is the solution gradient via MaxDO on only new parallel tasks, and \( \mathbf{g}_{\text{re}} \) means the gradient from the rehearsal data streams. The result shows that it is necessary to put the rehearsal gradient to the MaxDO. Otherwise, the model will get worse accuracy and forgetting.

### 4.5. Comparison with Symmetric Metrics

As shown in Table 3(b), we compare AGD with three common symmetric metrics including EuDist, CosDist, and Normalized EuDist. EuDist, CosDist are defined in Sec. 3.2. The vanilla EuDist depends highly on the gradient magnitude difference, thus we also compare with its normalized version \( D(x, y) = \frac{\|x - y\|}{\|x\| + \|y\|} \in [0, 1] \), namely normalized EuDist (Norm EuDist). The results show that the three metrics can also obtain good performance with MaxDO. However, EuDist fails to reduce catastrophic forgetting effectively because of the over-emphasizing of gradient magnitude difference. Considering only the gradient direction difference, MaxDO
Table 3. Rehearsal experiments, comparisons with symmetric metrics and ablation studies in MaxDO on PS-CIFAR-100.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Method</th>
<th>Task-increment</th>
<th>Class-increment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$A_e$ (%)</td>
<td>$F_e$ (%)</td>
</tr>
<tr>
<td>(a) Rehearsal</td>
<td>MaxDO (w/o Rehearsal)</td>
<td>25.241 ± 0.466</td>
<td>3.128 ± 0.175</td>
</tr>
<tr>
<td></td>
<td>MaxDO (w/o Rehearsal gradient)</td>
<td>47.653 ± 1.280</td>
<td>23.538 ± 1.468</td>
</tr>
<tr>
<td>(b) Symmetric metrics</td>
<td>MaxDO (EuDist)</td>
<td>49.201 ± 0.410</td>
<td>24.112 ± 0.197</td>
</tr>
<tr>
<td></td>
<td>MaxDO (CosDist)</td>
<td>49.154 ± 0.570</td>
<td>25.094 ± 0.222</td>
</tr>
<tr>
<td></td>
<td>MaxDO (Norm EuDist)</td>
<td>48.517 ± 0.336</td>
<td>24.386 ± 0.388</td>
</tr>
<tr>
<td>(c) Ablation studies</td>
<td>MGDA [15]</td>
<td>48.957 ± 0.451</td>
<td>25.088 ± 0.203</td>
</tr>
<tr>
<td></td>
<td>MaxDO (w/o Max-Discrepancy)</td>
<td>47.899 ± 1.022</td>
<td>23.874 ± 0.447</td>
</tr>
<tr>
<td></td>
<td>MaxDO (w/o MGDA)</td>
<td>49.631 ± 0.431</td>
<td><strong>25.816 ± 0.561</strong></td>
</tr>
<tr>
<td>(d) Our full method</td>
<td>MaxDO (AGD, w/ Rehearsal)</td>
<td><strong>50.203 ± 0.274</strong></td>
<td>24.510 ± 0.092</td>
</tr>
</tbody>
</table>

4.6. Ablation Study

We evaluate the impact of the two main components of MaxDO in Table 3(c). First, we block the maximum discrepancy in MaxDO (MaxDO (w/o Max-Discrepancy)), which means that we solve the min-distance problem with Eq. (4) directly. Because of the self-interference, the solution combines the minimum discrepancy but fails to effectively reduce the discrepancy from other gradients (47.899% and 10.441% for $A_e$). We then block the MGDA that obtains a weighted gradient not damage any tasks. MGDA is quite useful in traditional MTL tasks but is not suitable in PCL (w/o MGDA, 49.693% and 11.881% for $A_e$). Because of the diverse training process of parallel tasks, gradients are with large magnitude differences and MGDA prefers to set large factors to small gradients. We solve the problem by both MGDA and the maximum discrepancy, and the whole MaxDO method with AGD outperforms the two ablated methods (50.203% and 12.231% for $A_e$), where the characters of the two components are combined.

4.7. Procedure Time

In Fig. 5, we show the training time comparison on PS-CIFAR-100. We first compare the training time for 2 to 5 parallel tasks in one iteration (Fig. 5(a)). We find that the generation of task numbers will grow the training time, and MaxDO needs more time than other methods because multiple minimum distance optimizations are performed. Then, we show both the final accuracy and whole training time in Fig. 5(b). In the whole timeline, MaxDO gets slightly longer training time than other methods but better performance.

4.8. Tolerance Analysis in AGD

In our paper, AGD is designed to have a tolerance $\frac{1}{2}$ in Corollary 1. This is because updating with a zero gradient will neither improve nor damage the performance. Even though, we prefer positive influence rather than non-influence. Thus, we define that the distance $\widehat{D}(g_A, g_B)$ in the situation $\|g_A\| \ll \|g_B\|$ is the mid-level in the value range. Therefore, we study to change the tolerance and observe the performance change. The tolerance can be controlled by adding a factor $\gamma > 0$ at the denominator. Omitting the edge situation, we have

$$\widehat{D}_\gamma(g_A, g_B) = \frac{\|g_A - g_B\|}{\gamma \|g_B\| + \|g_A - g_B\|}.$$  

The experiments on different tolerances are shown in Table 4. The results show either larger or smaller tolerances compared to $\frac{1}{2}$ will get the performance drop during the PCL training.

5. Conclusion

In this paper, we studied to address the training conflict and catastrophic forgetting issues in Parallel Continual Learning (PCL). We presented that the two issues are rooted in the gradient discrepancies and formulated the problem into a minimum distance optimization among gradients. However, the distance metric is often set to be symmetric, which
is problematic in gradient descent. To evaluate the gradient discrepancy in PCL, we proposed an explicit Asymmetric Gradient Distance (AGD), which considers both gradient magnitude ratios and directions and has a tolerance when updating with a small gradient of inverse direction. Moreover, we proposed a novel Maximum Discrepancy Optimization (MaxDO) strategy to minimize the maximum discrepancy among multiple gradients and avoid self-interference. Solving by MaxDO with AGD, the parallel training in PCL reduces the influence of the training conflict and slows the catastrophic forgetting. We verified the proposed method on three image recognition datasets. The experimental results significantly showed the advantage of MaxDO and the effectiveness of the proposed AGD. We list the latent limitation of our method: (1) The MaxDO cannot guarantee a theoretical Pareto optimum in the training process like MGDA, which means a better trade-off can be obtained in the future. (2) The MaxDO method needs more time for training.

Acknowledgement

We gratefully acknowledge the invaluable support from the following funding bodies: the National Natural Science Foundation of China (62072334, 62276182), and the National Key Research and Development Program of China (2020YFC1522701).

References


[27] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2017. 1


[34] Andreas Maurer, Massi Pontil, and Bernardino Romera-Paredes. Sparse coding for multitask and transfer learning. In *ICML*, 2013. 2


[38] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *TPAMI*, 2018. 1, 2


