Beyond Not-Forgetting: Continual Learning with Backward Knowledge Transfer

Sen Lin
School of ECEE
Arizona State University
slin70@asu.edu

Li Yang
School of ECEE
Arizona State University
lyang166@asu.edu

Deliang Fan
School of ECEE
Arizona State University
dfan@asu.edu

Junshan Zhang
Department of ECE
University of California, Davis
jazh@ucdavis.edu

Abstract

By learning a sequence of tasks continually, an agent in continual learning (CL) can improve the learning performance of both a new task and 'old' tasks by leveraging the forward knowledge transfer and the backward knowledge transfer, respectively. However, most existing CL methods focus on addressing catastrophic forgetting in neural networks by minimizing the modification of the learnt model for old tasks. This inevitably limits the backward knowledge transfer from the new task to the old tasks, because judicious model updates could possibly improve the learning performance of the old tasks as well. To tackle this problem, we first theoretically analyze the conditions under which updating the learnt model of old tasks could be beneficial for CL and also lead to backward knowledge transfer, based on the gradient projection onto the input subspaces of old tasks. Building on the theoretical analysis, we next develop a ContinUal learning method with Backward knowlEdge tRansfer (CUBER), for a fixed capacity neural network without data replay. In particular, CUBER first characterizes the task correlation to identify the positively correlated old tasks in a layer-wise manner, and then selectively modifies the learnt model of the old tasks when learning the new task. Experimental studies show that CUBER can even achieve positive backward knowledge transfer on several existing CL benchmarks for the first time without data replay, where the related baselines still suffer from catastrophic forgetting (negative backward knowledge transfer). The superior performance of CUBER on the backward knowledge transfer also leads to higher accuracy accordingly.

1 Introduction

One ultimate goal of artificial intelligence is to build an agent that can continually learn a sequence of different tasks, so as to echo the remarkable learning capability of human beings during their lifespan. With more tasks being learnt, the agent is expected to be able to learn a new task more easily by leveraging the accumulated knowledge from old tasks (forward knowledge transfer), and also further improve the learning performance of old tasks based on the gained knowledge of related new tasks (backward knowledge transfer). Such a learning paradigm is known as continual learning (CL) [5, 23], which has recently attracted much attention.

Most existing CL methods focus on addressing the catastrophic forgetting problem [20], i.e., the neural network may easily forget the knowledge of old tasks when learning a new task. The main

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

https://doi.org/10.52202/068431-1176

strategy therein is to avoid the model change of old tasks when learning the new task. For example, regularization-based methods (e.g., [12, 1, 18]) penalize the modification of important weights of old tasks; parameter-isolation based methods (e.g., [7, 26, 31, 9]) fix the model learnt for old tasks; and memory-based methods (e.g., [3, 6, 25]) aim to update the model with minimal interference introduced to old tasks. While effectively mitigating catastrophic forgetting, such a model-freezing strategy inevitably limits the backward knowledge transfer, by implicitly and conservatively treating the model update of the new task as interference to old tasks. Intuitively, careful modifications of the learnt model of old tasks may further improve the learning performance especially when the new task shares similar knowledge to the old tasks.

This work seeks to explore CL from a new perspective by going beyond merely addressing forgetting: *When and how could we improve the learnt model of old tasks to facilitate backward knowledge transfer?* In particular, we consider a fixed neural network and the scenario where the data of old tasks is not accessible when learning a new task, which naturally rules out the expansion-based methods (e.g., [31, 9]) and the experience-replay methods (e.g., [22, 4]). Clearly, achieving backward knowledge transfer is very challenging in this case as mitigating forgetting herein is already nontrivial. To tackle this challenge, note that the orthogonal-projection based CL methods (e.g., [25, 17]), which update the model with gradients orthogonal to the input subspaces of old tasks, have demonstrated the state-of-the-art performance under the setting above. Thus motivated, we propose a new orthogonal-projection based CL method to carefully modify the learnt model of old tasks for better backward knowledge transfer.

The main contributions of this work include 1) the quantification of the conditions under which modifying the learnt model of old tasks is beneficial and 2) a new CL method inspired by the analysis, so as to answer the question above. More specifically, we first introduce notions of 'sufficient projection' and 'positive correlation' based on the gradient projection onto the subspaces of old tasks to characterize the task correlation. We theoretically show that when the task gradients are sufficiently aligned in the old task subspace, appropriately updating the learnt model of old tasks when learning the new task could improve the learning performance and possibly lead to a better model for the old tasks. Based on this analysis, we next propose an orthogonal-projection based ContinUal learning method with Backward knowlEdge tRansfer (CUBER), which 1) first identifies the positively correlated old tasks for the new task and 2) carefully updates the learnt model of selected old tasks along with the model learning for the new task. The experimental results on standard CL benchmarks show that CUBER can achieve the best backward knowledge transfer compared to all the considered baseline CL methods, which consequently improves the overall learning performance. In particular, to the best of our knowledge, CUBER is the first to demonstrate positive backward knowledge transfer on these benchmarks using a fixed capacity network without experience replay, whereas all the compared baselines still suffer from catastrophic forgetting with negative backward knowledge transfer.

2 Related work

Continual learning. Existing CL methods can be generally divided into three categories: 1) *Regularization-based methods* (e.g., [12, 1, 15, 18]) add additional regularization in the loss function to penalize the model change of the old tasks when learning the new task. For example, Elastic Weight Consolidation (EWC) [12] regularizes the update on the important weights that are evaluated using the Fisher Information matrix; a recursive gradient optimization method is proposed in [18] to modify the gradient direction for the objective function regularized by the model change during CL. 2) *Parameter-isolation based methods* (e.g., [24, 16, 31, 30, 26, 29]) allocate different model parameters to different tasks, fix the parameters for old tasks by masking them out during the new task learning and expand the network when needed. For example, Additive Parameter Decomposition (APD) [30] proposes a hierarchical knowledge consolidation method to separate task-specific parameters and task-shared parameters, where the task-specific parameters are kept intact to address forgetting. 3) *Memory-based methods* can be further divided into experience-replay methods and orthogonal-projection based methods depending on if the data of old tasks is available for the new task. Experience-replay methods (e.g., [3, 22, 8]) store and replay the old tasks data when learning the new task, while orthogonal-projection based methods (e.g., [32, 6, 25, 17]) update the model in the orthogonal direction of old tasks without storing the raw data of old tasks. In particular, [10] proposes natural continual learning based on Bayesian learning to unify weight regularization and orthogonal gradient projection.

https://doi.org/10.52202/068431-1176

However, [10] leverages the knowledge of the current task in an implicit way to improve on the old tasks, whereas our method attempts more explicitly to update the model of old tasks whenever relevant new information is available.

Knowledge transfer. Most of CL methods focus on the catastrophic forgetting problem, and little attention has been put on the knowledge transfer across tasks in CL with neural networks. Progressive Network [24] considers the forward knowledge transfer but learns one model for each task. [17] proposes trust region gradient projection (TRGP) to facilitate forward knowledge transfer from correlated old tasks to the new task through a scaling matrix, but the model is still updated along the direction orthogonal to the input subspaces of old tasks to mitigate forgetting. [11] studies the knowledge transfer in a mixed sequence of similar and dissimilar tasks, but employs a complicated task similarity detection mechanism where two additional networks need to be trained first for each task before learning the new task.

3 When could we improve the learnt model of old tasks?

In continual learning, a sequence of tasks $\mathbb{T} = \{t\}_{t=1}^{T}$ arrives sequentially. For each task t, there is a dataset $\mathbb{D}^t = \{(\boldsymbol{x}_i^t, \boldsymbol{y}_i^t)\}_{i=1}^{N^t}$ with N^t sample pairs, which is sampled from some unknown distribution $\mathbb{P}^t$. In this work, we consider a fixed capacity neural network with weights $\boldsymbol{w}$. When learning a new task t, we only have access to the dataset $\mathbb{D}^t$ and no datapoints of old tasks are available. We further denote $\mathcal{L}(\boldsymbol{w}, \mathbb{D}^t) = \mathcal{L}_t(\boldsymbol{w})$ as the loss function for training, e.g., mean squared and cross-entropy loss, and $\boldsymbol{w}^t$ as the model after learning task t.

Intuitively, if the new task t has strong similarity with some old tasks, appropriate model update of the new task would not introduce forgetting of the similar old tasks, but can lead to a better model for these old tasks due to the backward knowledge transfer. To formally characterize this task similarity, we introduce the following conditions on the task gradients.

Definition 1 (Sufficient Projection)**.** *For any new task $t \in [1, T]$, we say it has sufficient gradient projection on the input subspace of old task $j \in [0, t-1]$ if for some $\epsilon_1 \in (0, 1)$*

$$\|\mathrm{Proj}_{S^j}(\nabla \mathcal{L}_t(\boldsymbol{w}^{t-1}))\|_2 \geq \epsilon_1 \|\nabla \mathcal{L}_t(\boldsymbol{w}^{t-1})\|_2.$$

Here Proj_{S^j} defines the projection on the input subspace S^j of task j: $\mathrm{Proj}_{S^j}(A) = AB^j(B^j)'$ for some matrix A and B^j is the bases for S^j. This definition of sufficient projection follows the same line of the trust region defined in [17], which implies that task t and j may have sufficient common bases between their input subspaces and hence are strongly correlated, because the gradient lies in the span of the input [33].

Definition 2 (Positive Correlation)**.** *For any new task $t \in [1, T]$, we say it has positive correlation with old task $j \in [0, t-1]$ if for some $\epsilon_2 \in (0, 1)$*

$$\langle \nabla \mathcal{L}_j(\boldsymbol{w}^j), \nabla \mathcal{L}_t(\boldsymbol{w}^{t-1}) \rangle \geq \epsilon_2 \|\nabla \mathcal{L}_j(\boldsymbol{w}^j)\|_2 \|\nabla \mathcal{L}_t(\boldsymbol{w}^{t-1})\|_2.$$

While sufficient projection suggests the possibly strong correlation between two tasks t and j, Definition 2 goes one step further by introducing the positive correlation between tasks, in the sense that the initial model gradient of task t is not conflicting with the model gradient when learning task j. In fact, it can be shown that the positive correlation implies the sufficient projection in Definition 1 for some $\epsilon_1 \geq \epsilon_2$. Note that both Definition 1 and 2 characterize the correlation based on the initial model $\boldsymbol{w}^{t-1}$, which allows the task correlation detection before learning the new task t.

For ease of exposition, consider the scenario with a sequence of two tasks 1 and 2. Let $\mathcal{F}(\boldsymbol{w}) = \mathcal{L}(\boldsymbol{w}, \mathcal{D}_1) + \mathcal{L}(\boldsymbol{w}, \mathcal{D}_2)$, $\boldsymbol{g}_1(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \mathcal{D}_1)$ and $\boldsymbol{g}_2(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \mathcal{D}_2)$. Given the model learnt for task 1 as $\boldsymbol{w}^1$, we consider the following two model update rules for learning task 2, *where only the data of task 2 is available*:

- (Rule #1): $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha[\boldsymbol{g}_2(\boldsymbol{w}_k) - \mathrm{Proj}_{S^1}(\boldsymbol{g}_2(\boldsymbol{w}_k))]$,
- (Rule #2): $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \boldsymbol{g}_2(\boldsymbol{w}_k)$.

Here $\boldsymbol{w}_0 = \boldsymbol{w}^1$ and $k \in [0, K-1]$. Rule #1 characterizes the model update of the orthogonal-projection based methods, where no interference is introduced to task 1 as the model change is orthogonal to the input subspace S^1 of task 1 [17] (and hence the learnt model of task 1 will not

https://doi.org/10.52202/068431-1176

be updated). In contrast, the vanilla gradient descent Rule #2 for learning task 2 will inevitably modify the learnt model $\boldsymbol{w}^1$ for task 1 unless $\mathrm{Proj}_{S^1}(\boldsymbol{g}_2(\boldsymbol{w}_k)) = 0$, i.e., the extreme case where both forgetting and knowledge transfer do not occur. In what follows, we evaluate the performance of these update rules.

Theorem 1. *Suppose that the loss $\mathcal{L}$ is B-Lipschitz and $\frac{H}{2}$-smooth. Let $\alpha < \min\left\{\frac{1}{H}, \frac{\gamma\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{HBK}\right\}$ and $\epsilon_2 \geq \frac{(2+\gamma^2)\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{4\|\boldsymbol{g}_2(\boldsymbol{w}_0)\|}$ for some $\gamma \in (0,1)$. We have the following results:*

(1) If $\mathcal{L}$ is convex, Rule #2 for task 2 converges to the optimal model $\boldsymbol{w}^* = \arg\min\ \mathcal{F}(\boldsymbol{w})$;

(2) If $\mathcal{L}$ is nonconvex, Rule #2 for task 2 converges to the first order stationary point, i.e.,

$$\min_k \|\nabla\mathcal{F}(\boldsymbol{w}_k)\|^2 < \frac{2}{\alpha K}[\mathcal{F}(\boldsymbol{w}_0) - \mathcal{F}(\boldsymbol{w}^*)] + \frac{4+\gamma^2}{2}\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|^2.$$

Theorem 1 indicates that updating the model with Rule #2 will lead to the convergence to the minimizer of the joint objective function $\mathcal{F}(\boldsymbol{w})$ in the convex case, and the convergence to the first order stationary point in the nonconvex case, when task 1 and 2 satisfy the positive correlation with $\epsilon_2 \geq \frac{(2+\gamma^2)\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{4\|\boldsymbol{g}_2(\boldsymbol{w}_0)\|}$. That is to say, Rule #2 not only results in a good model for task 2, but can also be beneficial for the joint learning of task 1 and 2. Note that since $\boldsymbol{w}_0$ is the learnt model of task 1, in general we have $\|\boldsymbol{g}_1(\boldsymbol{w}_0)\| < \|\boldsymbol{g}_2(\boldsymbol{w}_0)\|$. Proof of Theorem 1 can be found in Appendix A.

Theorem 2. *Suppose that the loss $\mathcal{L}$ is B-Lipschitz and $\frac{H}{2}$-smooth. We have the following results:*

(1) Let $\boldsymbol{w}^c$ and $\boldsymbol{w}^r$ be the model parameters after applying one update to some initial model $\boldsymbol{w}$ by using Rule #1 and Rule #2, respectively. Suppose $\alpha < \min\left\{\frac{1}{H}, \frac{\gamma\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{HBK}\right\}$, $\epsilon_1 \geq \sqrt{\frac{1+2\alpha H}{2+\alpha H}}$ and $\epsilon_2 \geq \frac{(2+\gamma^2)\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{4\|\boldsymbol{g}_2(\boldsymbol{w}_0)\|}$ for some $\gamma \in (0,1)$. It follows that $\mathcal{F}(\boldsymbol{w}^r) \leq \mathcal{F}(\boldsymbol{w}^c)$;

(2) Let $\boldsymbol{w}_k$ be the k-th iterate for task 2 with Rule #2. Suppose that $\langle \boldsymbol{g}_1(\boldsymbol{w}_0), \boldsymbol{g}_2(\boldsymbol{w}_i)\rangle \geq \epsilon_2\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|\|\boldsymbol{g}_2(\boldsymbol{w}_i)\|$ for $i \in [0, k-1]$ and $\alpha \leq \frac{4\epsilon_2\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{HBk^{1.5}}$. It follows that $\mathcal{L}_1(\boldsymbol{w}_k) \leq \mathcal{L}_1(\boldsymbol{w}^1)$.

Intuitively, the first part of Theorem 2 shows that updating with Rule #2 can achieve lower loss value compared to Rule #1 after one step gradient update when task 1 and 2 satisfy the sufficient projection with $\epsilon_1 \geq \sqrt{\frac{1+2\alpha H}{2+\alpha H}}$ and the positive correlation with $\epsilon_2 \geq \frac{(2+\gamma^2)\|\boldsymbol{g}_1(\boldsymbol{w}_0)\|}{4\|\boldsymbol{g}_2(\boldsymbol{w}_0)\|}$. If the positive correlation condition also holds for iterates of Rule #2 when learning the task 2, the second part of Theorem 2 indicates that updating the model indeed leads to a better model for task 1 with respect to $\mathcal{L}_1$. In a nutshell, when 1) the task 2 has sufficient gradient projection onto the subspace of task 1 and 2) the projected gradient is also aligned well with the gradient of task 1 in the subspace of task 1, updating the model along $\boldsymbol{g}_2$ will modify the learnt model $\boldsymbol{w}^1$ towards a favorable direction for CL and enable the backward knowledge transfer to task 1. Proof of Theorem 2 is in Appendix B.

It is worth noting that the conditions for Theorem 1 and 2 depend only on the initial model gradient $\boldsymbol{g}_2(\boldsymbol{w}^1)$ before learning task 2 and the gradient $\boldsymbol{g}_1(\boldsymbol{w}^1)$ when learning task 1, which are both easily accessible and calculated. Particularly, the positive correlation can be evaluated by only storing the gradient $\boldsymbol{g}_1(\boldsymbol{w}^1)$ instead of the data of task 1. In stark contrast, the task gradient correlation characterization in [3, 22] involves the gradient evaluation of old tasks with respect to the current model weight, and hence requires the data of old tasks when learning the new task.

4 Continual learning with backward knowledge transfer

Based on the theoretical analysis above, we next propose a continual learning method with backward knowledge transfer (CUBER), by selectively updating the learnt model of old tasks when learning the new task. In particular, CUBER works in a layer-wise manner: Given a L-layer network, CUBER first characterizes the task correlation for each layer, and then employs different strategies to learn the new task depending on the task correlation.

More specifically, denote the set of weights as $\boldsymbol{w} = \{\boldsymbol{w}_l\}_{l=1}^L$, where $\boldsymbol{w}_l$ is the layer-wise weight for layer l. Given the data input $\boldsymbol{x}_i^t$ for task t, let $\boldsymbol{x}_{l,i}^t$ be the input of layer l and $\boldsymbol{x}_{1,i}^t = \boldsymbol{x}_i^t$. Denote f as the operation of the network layer. The output $\boldsymbol{x}_{l+1,i}^t$ for layer l can be then computed as

https://doi.org/10.52202/068431-1176

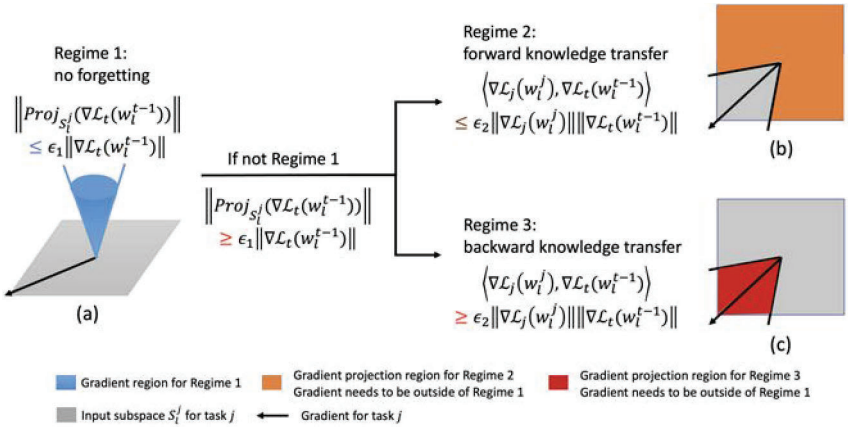

Figure 1: A simple illustration of the layer-wise task correlation detection. Given the new task t, an old task j belongs to (1) Regime 1 if the initial model gradient $\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1})$ of task t has small projection onto the subspace S_l^j of task j; (2) Regime 2 if the strong projection condition is satisfied while the projection of $\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1})$ onto S_l^j is not aligned well with the gradient of task j; (3) Regime 3 if both the strong projection condition and the positive correlation condition are satisfied.

$\boldsymbol{x}_{l+1,i}^t = f(\boldsymbol{w}_l, \boldsymbol{x}_{l,i}^t)$. Here we denote $\boldsymbol{x}_{l,i}^t$ as the representations of the input $\boldsymbol{x}_i^t$ at layer l. Given a new task $t \ge 2$, we characterize its task correlation with some old task $j \in [1, t-1]$ for layer l into three different regimes based on Definition 1 and 2.

Regime 1 (no forgetting): We say that task $j \in Reg_{l,1}^t$ for layer l if the following holds:

$$\|\mathrm{Proj}_{S_l^j}(\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1}))\| \le \epsilon_1\|\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1}))\|.$$

In this case, the layer-wise input subspace S_l^j for task j and S_l^t for task t are treated as nearly orthogonal ((a) in Fig. 1). As a result, there is little knowledge transfer between these two tasks, and updating the model along with $\nabla\mathcal{L}_t(\boldsymbol{w}_l)$ would not introduce much interference to task j. To reinforce the knowledge protection for task j, the model will be updated based on orthogonal projection:

$$\nabla\mathcal{L}_t(\boldsymbol{w}_l) \longleftarrow \nabla\mathcal{L}_t(\boldsymbol{w}_l) - \mathrm{Proj}_{S_l^j}(\nabla\mathcal{L}_t(\boldsymbol{w}_l)).$$

Regime 2 (forward knowledge transfer): We say that task $j \in Reg_{l,2}^t$ for layer l if the following holds:

$$\|\mathrm{Proj}_{S_l^j}(\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1}))\| \ge \epsilon_1\|\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1}))\|,$$
$$\langle\nabla\mathcal{L}_j(\boldsymbol{w}_l^j), \nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1})\rangle \le \epsilon_2\|\nabla\mathcal{L}_j(\boldsymbol{w}_l^j)\|\|\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1})\|.$$

In this case, task j and task t can be strongly correlated for layer l but possibly with 'negative' correlation ((b) in Fig. 1), in the sense that updating the model along with $\nabla\mathcal{L}_t(\boldsymbol{w}_l)$ would substantially modify the learnt model $\mathrm{Proj}_{S_l^j}(\boldsymbol{w}^{t-1})$ for task j in an unfavorable way and lead to the forgetting of task j. As there will be better forward knowledge transfer from the old task j to the new task t for layer l, we leverage the scaled weight projection in [17] to facilitate the forward knowledge transfer through a scaling matrix $Q_l^{j,t}$, whereas the model is still updated using orthogonal projection to protect the knowledge of old tasks:

$$\nabla\mathcal{L}_t(\boldsymbol{w}_l) \longleftarrow \nabla\mathcal{L}_t(\boldsymbol{w}_l) - \mathrm{Proj}_{S_l^j}(\nabla\mathcal{L}_t(\boldsymbol{w}_l)),$$
$$Q_l^{j,t} \longleftarrow Q_l^{j,t} - \beta\nabla_Q\mathcal{L}_t(\boldsymbol{w}_l - \mathrm{Proj}_{S_l^j}(\boldsymbol{w}_l) + \boldsymbol{w}_l B_l^j Q_l^{j,t}(B_l^j)'). \quad (1)$$

Here B_l^j is the bases matrix for subspace S_l^j. Intuitively, the scaled weight projection $\boldsymbol{w}_l B_l^j Q_l^{j,t}(B_l^j)'$ replaces the weight projection $\mathrm{Proj}_{S_l^j}(\boldsymbol{w}_l)$ of task j by a scaled version, which transforms the knowledge of task j to the appropriate model of the new task t through the optimization of $Q_l^{j,t}$.

Regime 3 (backward knowledge transfer): We say task $j \in Reg_{l,3}^t$ for layer l if the following holds:

$$\|\mathrm{Proj}_{S_l^j}(\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1}))\| \ge \epsilon_1\|\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1}))\|,$$
$$\langle\nabla\mathcal{L}_j(\boldsymbol{w}_l^j), \nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1})\rangle \ge \epsilon_2\|\nabla\mathcal{L}_j(\boldsymbol{w}_l^j)\|\|\nabla\mathcal{L}_t(\boldsymbol{w}_l^{t-1})\|.$$

https://doi.org/10.52202/068431-1176

With the sufficient projection and the positive correlation, updating the model along with $\nabla\mathcal{L}_t(\boldsymbol{w}_l)$ could possibly lead to a better model for continual learning and also improve the learning performance of the old task j ((c) in Fig. 1). To avoid overly-optimistic modification on the learnt model of task j, we further regularize the projection of the model change on the subspace S_l^j, given that the model projection is indeed frozen for task j to address forgetting with orthogonal projection, i.e., $\text{Proj}_{S_l^j}(\boldsymbol{w}_l^{t-1}) = \text{Proj}_{S_l^j}(\boldsymbol{w}_l^j)$. This gives the following model update:

$$\boldsymbol{w}_l \longleftarrow \boldsymbol{w}_l - \alpha\nabla[\mathcal{L}_t(\boldsymbol{w}_l) + \lambda\|\text{Proj}_{S_l^j}(\boldsymbol{w}_l - \boldsymbol{w}_l^{t-1})\|].$$

Note that the gradient projection on the subspaces of old tasks in Regime 1 and 2 is removed from the gradient in the model update above. Besides, we also learn a scaling matrix $Q_l^{j,t}$ for better forward knowledge transfer from the old task $j \in Reg_{l,3}^t$ to the new task t as in Eq. (1).

However, continuous model update with task t gradient $\nabla\mathcal{L}_t(\boldsymbol{w}_l)$ will eventually lead to the task specific model for task t, which usually differs from the model of task j in Regime 3 (Fig. 2). To address this problem, note that the second part of Theorem 2 characterizes the condition under which updating the model with $\nabla\mathcal{L}_t(\boldsymbol{w}_l)$ will result in backward knowledge transfer. Thus motivated, for a model update iterate $\boldsymbol{w}_{l,k}$ at k-th iteration when learning the new task t, we evaluate the following condition for task $j \in Reg_{l,3}^t$:

$$\langle\nabla\mathcal{L}_j(\boldsymbol{w}_l^j), \nabla\mathcal{L}_t(\boldsymbol{w}_{l,k})\rangle \geq \epsilon_2\|\nabla\mathcal{L}_j(\boldsymbol{w}_l^j)\|\|\nabla\mathcal{L}_t(\boldsymbol{w}_{l,k})\| \quad (2)$$

and degenerate task j to Regime 2, i.e., remove the gradient projection on the subspace S_l^j from the task t gradient and stop modifying the model for task j, if the condition (2) does not hold.

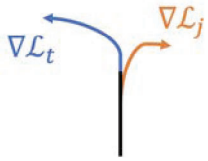

Figure 2: A simple illustration of the case where updating the model with $\nabla\mathcal{L}_t$ benefits the old task $j \in Reg_{l,3}^t$ at the beginning but eventually conflicts with $\nabla\mathcal{L}_j$.

Bases extraction: After learning the model $\boldsymbol{w}^t$ for task t, we construct the input subspace for each layer l by extracting bases from its representation $\boldsymbol{x}_{l,i}^t$ based on singular value decomposition (SVD) [17]. More specifically, given a batch of n samples and the learnt model $\boldsymbol{w}^t$, the representation matrix for layer l is denoted as $R_l^t = [\boldsymbol{x}_{l,1}^t, \boldsymbol{x}_{l,2}^t, ..., \boldsymbol{x}_{l,n}^t]$. Since the bases of the old tasks may include important bases for the new task t, we determine the bases for task t from the union of the bases of the old tasks and the newly generated bases. Towards this end, we first concatenate the bases B_l^j for $j \in [0, t-1]$ together in a matrix O_l^t and remove the common bases. Then SVD is applied on the residual representation matrix $\tilde{R}_l^t = R_l^t - R_l^t O_l^t (O_l^t)'$, i.e., $\tilde{R}_l^t = U_l^t \Sigma_l^t (V_l^t)'$, where U_l^t is the left singular matrix. We construct B_l^t by selecting the most important bases from the pool of bases in O_l^t and U_l^t depending on their eigenvalues, which yields a low rank matrix approximation of R_l^t.

To conclude, the optimization problem for learning the new task t can be summarized as follows:

$$\begin{aligned}
\min_{\boldsymbol{w}, \{Q_l^{j,t}\}_{l, j\in Reg_{l,2}^t \cup Reg_{l,3}^t}} \quad & \mathcal{L}_t(\{\tilde{\boldsymbol{w}}_l\}_l) + \lambda \sum_l \sum_{j\in Reg_{l,3}^t} \|\text{Proj}_{S_l^j}(\boldsymbol{w}_l - \boldsymbol{w}_l^{t-1})\|, \quad (3)\\
s.t. \quad & \tilde{\boldsymbol{w}}_l = \boldsymbol{w}_l + \sum_{j\in Reg_{l,2}^t \cup Reg_{l,3}^t} [\boldsymbol{w}_l B_l^j Q_l^{j,t} (B_l^j)' - \text{Proj}_{S_l^j}(\boldsymbol{w}_l)],\\
& \nabla\mathcal{L}_t(\boldsymbol{w}_l) = \nabla\mathcal{L}_t(\boldsymbol{w}_l) - \sum_{j\in Reg_{l,1}^t \cup Reg_{l,2}^t} \text{Proj}_{S_l^j}(\nabla\mathcal{L}_t(\boldsymbol{w}_l)).
\end{aligned}$$

The key idea is that we conservatively update the model for old tasks in Regime 3 while using orthogonal projection to preserve the knowledge of other old tasks; in the meanwhile, we leverage the scaled weight projection to reuse the model knowledge of old tasks in both Regime 2 and 3 to facilitate forward knowledge transfer. It is worth to note that the task correlation is determined before learning the new task t, as both the strong projection condition and the positive correlation condition only depend on the initial model gradient for the new task. And this can be achieved by a simple forward-backward pass through the initial model with a batch of new task data. The overview of CUBER can be found in Algorithm 1.

https://doi.org/10.52202/068431-1176

Algorithm 1 Continual learning with backward knowledge transfer (CUBER)

Input: task sequence $\mathbb{T} = \{t\}_{t=1}^{T}$;
Learn the first task using vanilla stochastic gradient descent;
Extract the bases $\{B_l^1\}$ based on SVD using the learnt model $\boldsymbol{w}^1$;
for each task t **do**
    Calculate gradient $\nabla\mathcal{L}_t(\boldsymbol{w}_{t-1})$;
    Evaluate the strong projection and the positive correlation conditions for layer-wise task correlation detection to determine $Reg_{l,1}^t$, $Reg_{l,2}^t$ and $Reg_{l,3}^t$;
    for k=1, 2,... **do**
        Update the model and scaling matrices by solving Eq. (3);
        for task $j < t$ and $j \in Reg_{l,3}^t$ **do**
            if $\langle \nabla\mathcal{L}_j(\boldsymbol{w}_l^j), \nabla\mathcal{L}_t(\boldsymbol{w}_{l,k})\rangle < \epsilon_2 \|\nabla\mathcal{L}_j(\boldsymbol{w}_l^j)\| \|\nabla\mathcal{L}_t(\boldsymbol{w}_{l,k})\|$ **then**
                Degenerate task j to $Reg_{l,2}^t$;
            end if
        end for
    end for
    Store the gradient $\nabla\mathcal{L}_t(\boldsymbol{w}^t)$ in the task memory;
    Extract the bases $\{B_l^t\}$ based on SVD using the learnt model $\boldsymbol{w}^t$;
end for

5 Experiments

Datasets. We evaluate the performance of CUBER on four standard CL benchmarks. (1) Permuted MNIST: a variant of the MNIST dataset [14] where random permutations are applied to the input pixels. Following [19, 25], we divide the dataset into 10 tasks with different permutations and each task includes 10 classes. (2) Split CIFAR-100: we divide the CIFAR-100 dataset [13] into 10 different tasks, where each task is a 10-way multi-class classification problem. (3) 5-Datasets: we consider a sequence of 5 datasets, i.e., CIFAR-10, MNIST, SVHN [21], not-MNIST[2], Fashion MNIST[28], and the classification problem on each dataset is a task. (4) Split MiniImageNet: we divide the MiniImageNet dataset [27] into 20 tasks, where each task includes 5 classes.

Baselines. In this work, we compare CUBER with the following baselines on the benchmarks mentioned above. (1) EWC [12]: a regularization-based method that leverages Fisher Information matrix for weights importance evaluation; (2) HAT [26]: learns a hard attention mask to preserve the knowledge of old task; (3) Orthogonal Weight Modulation (OWM) [32]: learns a projector matrix to project the gradient of the new task to the orthogonal direction of input subspace of old tasks; (4) Gradient Projection Memory (GPM) [25]: stores the bases of the input subspace of old tasks and then updates the model with the gradient projection orthogonal to the subspace spanned by these bases; (5) TRGP [17]: proposes a scaled weight projection to facilitate the forward knowledge transfer from related old tasks to the new task while updating the model based on orthogonal gradient projection, which demonstrates the state-of-the-art performance for a fixed capacity network; (6) Averaged GEM (A-GEM) [3]: constrains the new task learning with the gradient calculated using the stored data of old tasks; (7) Experience Replay with Reservior sample (ER_Res) [4]: uses a small episodic memory to store old task samples for addressing forgetting; (8) Multitask: jointly learns all tasks once with a single network using all datasets, which usually serves as a performance upper bound in CL [25].

Network and training details. For a given dataset, we study all CL methods using the same network architecture. More specifically, for Permuted MNIST, we consider a 3-layer fully-connected network including 2 hidden layers with 100 units. And we train the network for 5 epochs with a batch size of 10 for every task. For Split CIFAR-100, we use a version of 5-layer AlexNet by following [25, 17]. When learning each task, we train the network for a maximum of 200 epochs with early termination based on the validation loss, and use a batch size of 64. For 5-Datasets, we use a reduced ResNet-18 [19] and follow the same training strategy as in Split CIFAR-100. For Split MiniImageNet, a reduced ResNet-18 is also used, and we train the network for a maximum of 100 epoches with early termination. The batch size is 64. Similar to [17], we select at most two tasks to be in Regime 2 and 3 for each layer with the largest gradient projection norm, to reduce the performance sensitivity on the choice of ϵ_1. In the experiments, we set $\epsilon_1 = 0.5$.

To evaluate the learning performance, we consider the following two metrics, i.e., accuracy (ACC) which measures the final accuracy averaged over all tasks, and backward transfer (BWT) which

https://doi.org/10.52202/068431-1176

measures the average accuracy change of each task after learning new tasks:

$$ACC = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}, \;\; BWT = \frac{1}{T-1}\sum_{i=1}^{T-1}(A_{T,i} - A_{i,i})$$

where $A_{i,j}$ represents the testing accuracy of task j after learning task i.

5.1 Main results

As shown in Table 1, CUBER demonstrates the best performance of BWT on all datasets compared to the baselines. In particular, *positive BWT can be obtained by CUBER on Split CIFAR-100 and Split MiniImageNet, which has not been achieved by previous works in a fixed capacity network without data-replay*. For 5-Dataset, since the tasks therein are less related to each other as in GPM [25], we do not expect much knowledge transfer across tasks (but knowledge transfer still exists in terms of the layer-wise features), and there is no forgetting in CUBER. Achieving zero-forgetting is very difficult for Permuted MNIST, because all the tasks share one output layer and there is no task identifier during testing (domain-incremental) [25]. However, even in this case CUBER still achieves the best BWT, i.e, nearly non-forgetting, among all methods. Clearly, the strong performance on BWT indicates that CUBER can effectively facilitate the backward knowledge transfer by wisely modifying the learnt model of old tasks.

Benefiting from the superior performance on the backward knowledge transfer, CUBER also achieves the best or comparable performance on the averaged accuracy. More specifically, CUBER improves around 1% in ACC over the best prior results on Split CIFAR-100, Split MiniImageNet and Permuted MNIST, while showing the comparable performance with the state-of-the-art method TRGP even on 5-datasets where tasks are less correlated. Moreover, CUBER performs better than Multitask on both 5-Dataset and Permuted MNIST, *which implies the importance of studying knowledge transfer in CL:* Both TRGP and CUBER outperform Multitask on 5-Dataset because of the scaled weight projection to facilitate forward knowledge transfer, whereas by facilitating backward knowledge CUBER becomes the only method that outperforms Multitask on Permuted MNIST.

Table 1: The ACC and BWT with the standard deviation values over 5 different runs on different datasets. Here for Split CIFAR-100, Split MiniImageNet and 5-Dataset we use a multi-head network, while a single-head network is used for Permuted MNIST. Moreover, $\epsilon_2 = 0.0$.

Method	Multi-head						Domain-incremental	
	Split CIFAR-100		Split MiniImageNet		5-Dataset		Permuted MNIST	
	ACC(%)	BWT(%)	ACC(%)	BWT(%)	ACC(%)	BWT(%)	ACC(%)	BWT(%)
Multitask	79.58 ± 0.54	-	69.46 ± 0.62	-	91.54 ± 0.28	-	96.70 ± 0.02	-
OWM	50.94 ± 0.60	-30 ± 1	-	-	-	-	90.71 ± 0.11	-1 ± 0
EWC	68.80 ± 0.88	-2 ± 1	52.01 ± 2.53	-12 ± 3	88.64 ± 0.26	-4 ± 1	89.97 ± 0.57	-4 ± 1
HAT	72.06 ± 0.50	0 ± 0	59.78 ± 0.57	-3 ± 0	91.32 ± 0.18	-1 ± 0	-	-
A-GEM	63.98 ± 1.22	-15 ± 2	57.24 ± 0.72	-12 ± 1	84.04 ± 0.33	-12 ± 1	83.56 ± 0.16	-14 ± 1
ER_Res	71.73 ± 0.63	-6 ± 1	58.94 ± 0.85	-7 ± 1	88.31 ± 0.22	-4 ± 0	87.24 ± 0.53	-11 ± 1
GPM	72.48 ± 0.40	-0.9 ± 0	60.41 ± 0.61	-0.7 ± 0.4	91.22 ± 0.20	-1 ± 0	93.91 ± 0.16	-3 ± 0
TRGP	74.46 ± 0.32	-0.9 ± 0.01	61.78 ± 0.60	-0.5 ± 0.6	$\mathbf{93.56} \pm 0.10$	-0.04 ± 0.01	96.34 ± 0.11	-0.8 ± 0.1
CUBER (ours)	$\mathbf{75.54} \pm 0.22$	$\mathbf{0.13} \pm 0.08$	$\mathbf{62.67} \pm 0.74$	$\mathbf{0.23} \pm 0.15$	$\mathbf{93.48} \pm 0.10$	$\mathbf{0.00} \pm 0.02$	$\mathbf{97.25} \pm 0.00$	$\mathbf{-0.02} \pm 0.00$

5.2 Ablation studies

Table 2: The comparison between CUBER and TRGP in OL-CIFAR100. The selected old task (*) in (b) represents the old task with the largest number of layers in Regime 3 of the new task.

(a) The comparison of ACC and BWT.

Method	ACC(%)	BWT(%)
CUBER	74.94	0.28
TRGP	74.33	-0.18

(b) BWT-S of the selected old tasks.

	Task 1	Task 2	Task 4	Task 5	Average
selected old task*	Task 0	Task 1	Task 3	Task 0	-
BWT-S (CUBER)	0.50	0.71	0.03	0.07	0.33
BWT-S (TRGP)	-0.20	0.10	-0.60	-0.20	-0.23

Backward knowledge transfer. The value of backward knowledge transfer indicates the average accuracy improvement for each old task after learning all tasks, which implies that the new task learning provides additional useful information for learning features of similar old tasks. Intuitively, this value should depend on the task similarity in CL. To better understand the advantage of CUBER

in terms of backward knowledge transfer, we further consider a special setup which includes a sequence of similar and dissimilar tasks. Specifically, different with Split-CIFAR100 where no tasks have overlapping classes, we split the first 50 classes in CIFAR100 into 7 tasks (OL-CIFAR100): Task 0-6 contain classes 0-9, 5-14, 10-19, 20-29, 25-34, 30-39, 40-49, respectively. We compare the performance of CUBER with TRGP in Table 2. As shown in Table 2a, CUBER clearly outperforms TRGP in both ACC and BWT. In particular, CUBER has a positive BWT of 0.28%, where TRGP suffers from forgetting. We further analyze the task selections in Table 2b, where "selected old task" refers to the task that has the largest number of layers in Regime 3 of the new task. For example, according to the setup of OL-CIFAR100, the selected old task for Task 4 should be Task 3 with a high probability as they have overlapping classes. And we denote a new metric, namely BWT-S, to evaluate the backward knowledge transfer of the selected old task after learning the new task, i.e., BWT-S=$A_{t,j} - A_{t-1,j}$ where j is the "selected old task" of the new task t. As shown in Table 2b, CUBER correctly identifies the correlated tasks for most new tasks except Task 5. This is a reasonable result because the task correlation detection is based on the initial model gradient which is noisy in general, and hence only serves as an estimation of underlying true correlation. However, such an estimated task correlation characterization is indeed sufficient to effectively facilitate backward knowledge transfer, as corroborated by the positive BWT-S and the superior performance of CUBER.

Forward knowledge transfer. We also evaluate the forward knowledge transfer (FWT) in CUBER, compared to the best two baseline methods GPM and TRGP. Here the FWT measures the gap between $A_{i,i}$ and the accuracy of learning task i only from scratch. For simplicity, we use the FWT of GPM as a baseline and evaluate the improvements of TRGP and CUBER over GPM. It can be seen from Table 3 that CUBER performs even better than TRGP although CUBER follows the same strategy, i.e., the scaled weight projection, to facilitate the forward knowledge transfer, and achieves the best FWT among the three methods in most cases. The reason behind is that the characterization of Regime 3 in CUBER not only allows the modification of the learnt model of the old tasks to prompt the backward knowledge transfer, but also relaxes the constraint on the gradient update for the new task, i.e., the model can be now updated in the subspace of the selected old tasks for learning the new task. This gradient constraint relaxation consequently leads to better model learning of the new task.

Table 3: Comparison of FWT among GPM, TRGP and CUBER. The value for GPM is zero because we treat GPM as the baseline and consider the relative FWT improvement over GPM.

Method	Split CIFAR-100	Split MiniImageNet	5-Dataset	Permuted MNIST
GPM	0	0	0	0
TRGP	2.01	2.36	1.98	0.18
CUBER	2.79	3.13	1.96	0.8

Impact of ϵ_2. It is clear that the selection of layer-wise Regime 3 depends on the value of the threshold ϵ_2. To show the impact of ϵ_2, we evaluate the learning performance under different values of ϵ_2 in Split CIFAR-100. As shown in Table 4, the performance on ACC is comparable for all three cases and the BWT decreases as the value of ϵ_2 increases. Intuitively, ϵ_2 characterizes the conservatism in selecting tasks to Regime 3 and modifying the model of selected tasks. Specifically, with a larger ϵ_2, we just consider the backward knowledge transfer to the old tasks that are strongly correlated with the new task, and only slightly modify the learnt model of these selected tasks, because the condition Eq. (2) can be quickly violated with the model update and CUBER will stop changing the learnt model of the selected tasks.

Table 4: The impact of ϵ_2 on the performance in Split CIFAR-100.

$\epsilon_2 = 0.0$		$\epsilon_2 = 0.2$		$\epsilon_2 = 0.5$	
ACC(%)	BWT(%)	ACC(%)	BWT(%)	ACC(%)	BWT(%)
75.54	0.22	75.73	0.03	75.55	0.01

6 Conclusion

In this work, we study the problem of backward knowledge transfer in CL. Different from most existing methods that generally freeze the learnt model of the old tasks so as to mitigate catastrophic forgetting, this study seeks to carefully modify the learnt model to facilitate backward knowledge transfer from the new task to the old tasks. To this end, we first introduce notions of strong projection

https://doi.org/10.52202/068431-1176

and positive correlation to characterize the task correlation, and show that when the task gradients are sufficiently aligned in the old task subspace, appropriate model change for the old tasks could be beneficial for CL and result in better backward knowledge transfer. Based on the theoretical analysis, we next propose CUBER to carefully learn the model for the new task, which would carefully update the learnt model of the old tasks that are positively correlated with the new task. As shown in the experimental results, CUBER can successfully improve the backward knowledge transfer on the standard CL benchmarks in contrast to related baselines.

Impact and limitations. The mainstream strategy nowadays to address forgetting is to minimize the interference to old tasks and avoid the learnt model change, which may however conflict with the goal of CL, in the sense that the backward knowledge transfer from the new task to old tasks can be restricted without modifying the learnt model. In this work, we go beyond this strategy and shed light on the relationship between backward knowledge transfer and model modification, by characterizing the task correlations with gradient projection. We hope that this work will serve as initial steps and motivate further research in CL community on the important while less explored problem, i.e., how to design algorithms that can provide targeted treatments to achieve backward knowledge transfer. However, CUBER also comes with several limitations. As in recent orthogonal-projection based CL methods, CUBER extracts the bases of the task subspaces based on SVD, which may lead to high computational cost for large dimensional data. How to reduce the complexity is an interesting direction. Another limitation is that we assume that clear task boundaries exist between different tasks. In future work, it is of great interests to extend CUBER to more general CL settings.

Acknowledgement

The work of S. Lin and J. Zhang was supported in part by the U.S. National Science Foundation Grants CNS-2203239, CNS-2203412, RINGS-2148253, and CCSS-2203238. The work of L. Yang and D. Fan was supported in part by the U.S. National Science Foundation Grants No. 1931871 and No. 2144751.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[2] Yaroslav Bulatov. Notmnist dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: http://yaroslavvb. blogspot. it/2011/09/notmnist-dataset. html*, 2, 2011.

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

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

[5] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[6] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[7] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[8] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory based lifelong learning algorithm. In *Conference on Neural Information Processing Systems*, 2020.

[9] Steven CY Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. *arXiv preprint arXiv:1910.06562*, 2019.

[10] Ta-Chu Kao, Kristopher Jensen, Gido van de Ven, Alberto Bernacchia, and Guillaume Hennequin. Natural continual learning: success is a journey, not (just) a destination. *Advances in Neural Information Processing Systems*, 34:28067–28079, 2021.

[11] Zixuan Ke, Bing Liu, and Xingchang Huang. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in Neural Information Processing Systems*, 33:18493–18504, 2020.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

[15] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *arXiv preprint arXiv:1703.08475*, 2017.

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[17] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Trgp: Trust region gradient projection for continual learning. *Tenth International Conference on Learning Representations, ICLR 2022*, 2022.

[18] Hao Liu and Huaping Liu. Continual learning with recursive gradient optimization. *arXiv preprint arXiv:2201.12522*, 2022.

[19] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[20] 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. Elsevier, 1989.

[21] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[22] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[23] Mark Bishop Ring et al. Continual learning in reinforcement environments. 1994.

[24] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[25] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

[26] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

https://doi.org/10.52202/068431-1176

[27] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[28] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[29] Li Yang, Sen Lin, Junshan Zhang, and Deliang Fan. Grown: Grow only when necessary for continual learning. *arXiv preprint arXiv:2110.00908*, 2021.

[30] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *Eighth International Conference on Learning Representations, ICLR 2020*. ICLR, 2020.

[31] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[32] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[33] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Theorem 1 and 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the code in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See section 5.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See section 5.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the appendix.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See section 5.

(b) Did you mention the license of the assets? [Yes] See the appendix.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include code in the supplemental material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

https://doi.org/10.52202/068431-1176