



# Restricted orthogonal gradient projection for continual learning

Zeyuan Yang<sup>a</sup>, Zonghan Yang<sup>a</sup>, Yichen Liu<sup>c</sup>, Peng Li<sup>b,\*</sup>, Yang Liu<sup>a,b</sup>

<sup>a</sup> Department of Computer Science and Technology, Tsinghua University, China

<sup>b</sup> Institute for AI Industry Research, Tsinghua University, China

<sup>c</sup> College of Engineering, Peking University, China

## ARTICLE INFO

### Keywords:

Continual learning

Gradient projection

Forward knowledge transfer

## ABSTRACT

Continual learning aims to avoid catastrophic forgetting and effectively leverage learned experiences to master new knowledge. Existing gradient projection approaches impose hard constraints on the optimization space for new tasks to minimize interference, which simultaneously hinders forward knowledge transfer. To address this issue, recent methods reuse frozen parameters with a growing network, resulting in high computational costs. Thus, it remains a challenge whether we can improve forward knowledge transfer for gradient projection approaches *using a fixed network architecture*. In this work, we propose the Restricted Orthogonal Gradient prOjection (ROGO) framework. The basic idea is to adopt a restricted orthogonal constraint allowing parameters optimized in the direction oblique to the whole frozen space to facilitate forward knowledge transfer while consolidating previous knowledge. Our framework requires neither data buffers nor extra parameters. Extensive experiments have demonstrated the superiority of our framework over several strong baselines. We also provide theoretical guarantees for our relaxing strategy.

## 1. Introduction

A critical capability for intelligent systems is to continually learn given a sequence of tasks (Thrun and Mitchell, 1995; McCloskey and Cohen, 1989). Unlike human beings, vanilla neural networks straightforwardly update parameters regarding current data distribution when learning new tasks, suffering from catastrophic forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990; Kirkpatrick et al., 2017). Continual learning without forgetting has thus gained increasing attention in recent years (Kurle et al., 2019; Ehret et al., 2020; Liu and Liu, 2022). Moreover, an ideal continual learner is expected to not only avoid catastrophic forgetting but also facilitate forward knowledge transfer (Lopez-Paz and Ranzato, 2017), i.e., leveraging past learning experiences to master new knowledge efficiently and effectively (Parisi et al., 2019; Finn et al., 2019). Without forward knowledge transfer, approaches may have limited performance even with less forgetting (Kemker et al., 2018).

To mitigate forgetting and facilitate forward knowledge transfer, replay-based methods (Lopez-Paz and Ranzato, 2017; Shin et al., 2017; Choi et al., 2021) stores some old samples in the memory, and expansion-based methods (Rusu et al., 2016; Yoon et al., 2017, 2019) expand the model structure to accommodate incoming knowledge. However, these methods require either extra memory buffers (Parisi et al., 2019) or a growing network architecture as new tasks continually arrive (Kong et al., 2022), which are always computationally

expensive (De Lange et al., 2021). Thus, promoting performance within a fixed network capacity remains challenging. Regularization-based methods (Kirkpatrick et al., 2017; De Lange et al., 2021) penalize the transformation of parameters regarding the corresponding plasticity via regularization terms. Instead of constraining individual neurons explicitly, recent gradient projection methods (Zeng et al., 2019; Saha et al., 2021; Wang et al., 2021) further constrain the directions of gradient update, obtaining superior performance. However, in spite of effectively mitigating forgetting, the limited optimization space also hinders the capability of learning new tasks (Zeng et al., 2019), resulting in insufficient forward knowledge transfer.

Particularly, we illustrate the optimizing process of traditional gradient projection methods by the black dashed line in Fig. 1. As shown in Fig. 1, the strict orthogonal constraint may exclude the global optimum (the red star) from the optimization space. In other words, constraining the directions of gradient update fails on the plasticity in the stability-plasticity dilemma (French, 1997). TRGP (Lin et al., 2022) tackles this problem by allocating new parameters within the selected subspace of old tasks as trust regions, suffering a similar computational cost burden as expansion-based methods (Wang et al., 2021). Therefore, promoting forward knowledge transfer within a fixed network capacity remains a key challenge for gradient projection methods.

\* Corresponding author.

E-mail address: [lipeng@air.tsinghua.edu.cn](mailto:lipeng@air.tsinghua.edu.cn) (P. Li).

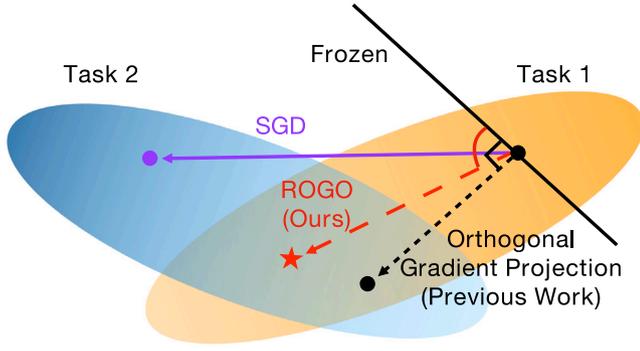


Fig. 1. Illustration of our ROGO framework. The lines with arrows denote the gradients on task 2 (pale blue area) after learning task 1 (pale yellow area). We indicated the global optimum by the red star (★). The black straight line denotes the frozen space, to which gradient projection methods constrain the gradients to be orthogonal.

To address this challenge, we propose our Restricted Orthogonal Gradient prOjection (ROGO) framework to facilitate forward knowledge transfer. Instead of optimizing in an orthogonal direction, our method adopts a restricted orthogonal constraint. As illustrated by the red line in Fig. 1, our framework allows the parameters to be updated in the direction oblique to the original frozen space, which explores a larger optimization space and thus obtains better performance on new tasks. Specifically, we design a simple yet effective strategy to find the critical subspace, namely the relaxing space, within the frozen space. The parameters are jointly optimized in the relaxing and original optimization spaces to facilitate forward knowledge transfer. A complimentary regularization term is introduced to consolidate previous knowledge as well. Extensive experiments on various continual learning benchmarks demonstrate that our ROGO framework promotes **forward knowledge transfer** and achieves better classification performance compared with related state-of-the-art approaches. Moreover, our framework can also be extended as an expansion-based method by storing the parameters of the selected relaxing space, universally surpassing TRGP (Lin et al., 2022) and other expansion-based approaches. We also provide theoretical proof to guarantee the efficiency of our strategy.

## 2. Related work

### 2.1. Gradient projection methods

Gradient projection methods constrain the gradients to be orthogonal to the frozen space constructed by previous tasks to overcome forgetting. OWM (Zeng et al., 2019) first proposed to modify the gradients upon projector matrices. OGD (Farajtabar et al., 2020) further keeps the gradients orthogonal to the space spanned by previous gradients, whereas GPM (Saha et al., 2021) computes the frozen space based on old data. NCL (Kao et al., 2021) combines the idea of gradient projection and Bayesian weight regularization to mitigate catastrophic forgetting. In spite of minimizing backward interference, these approaches suffer poor forward transfer and lack plasticity (Kong et al., 2022). FS-DGPM (Deng et al., 2021) introduces a soft weight to regulate the frozen space to achieve better performance on new tasks, requiring extra data buffer and over ten times the training time. TRGP (Lin et al., 2022) further expands the model with trust regions, introducing additional scale parameters. On the contrary, we focus on facilitating forward knowledge transfer *relying on either previous data or extra parameters* by optimizing parameters in the direction oblique to the frozen space.

### 2.2. Regularization-based methods

Regularization-based methods (Zhu et al., 2021; Yu et al., 2020) introduce regularization terms to the objective function to penalize the modification of parameters, requiring neither data buffers nor extra parameters as gradient projection methods. EWC (Kirkpatrick et al., 2017) first proposes to constrain the change based on approximated importance weights. HAT (Serra et al., 2018) learns task-based hard attention to identify important parameters. Other methods, also called parameter-isolation methods, defy forgetting via freezing the gradient updates of particular parameters (De Lange et al., 2021). PackNet (Mallya and Lazebnik, 2018) iteratively prunes and allocates parameters subset to incoming tasks, whereas Kumar et al. (2021) facilitate inter-task transfer with the IBP (Griffiths and Ghahramani, 2011). Instead of restricting individual parameter updates, our approach constrains the direction of gradients.

### 2.3. Other methods

Replay-based methods (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2018) maintain a complementary memory for old data, which are replayed during learning new tasks. Recent approaches (Chen-shen et al., 2018; Cong et al., 2020) deploy auxiliary deep generative models to synthesize pseudo data. However, including extra data into the current task introduces excessive training time (De Lange et al., 2021). Expansion-based methods (Yoon et al., 2017, 2019) dynamically allocate new parameters to learn new tasks. While these methods face capacity explosion inevitably after learning a long sequence of tasks. In contrast, our method maintains a fixed network architecture and requires no previous data.

## 3. Restricted orthogonal gradient projection

### 3.1. Preliminaries

In a continual learning setting, we consider  $T$  tasks arriving as a sequence. When learning the current task, the datasets of old tasks are inaccessible. We use an  $L$ -layer neural network with fixed capacity, and parameters defined as  $\mathcal{W} = \{W^l\}_{l=1}^L$ , where  $W^l$  denotes the parameters in the  $l$ th layer.

To mitigate catastrophic forgetting, gradient projection methods construct the representative space, namely the frozen space, based on previous tasks and optimize the parameters orthogonal to the frozen space. Specifically, for each task  $t$ , Saha et al. (2021) obtain the layer-wise representation space  $R_t^l$  by compressing the representation matrix  $\mathbf{H}_t^l = [h_{t,1}^l, \dots, h_{t,N_t}^l]$ , where  $N_t$  denotes the number of samples in task  $t$  and  $h_{t,j}^l$  denotes the intermediate representation of the  $j$ th input. The constructed representation spaces of previous tasks are then concatenated to be the frozen gradient space  $\mathcal{U}_t = \{U_t^l\}_{l=1}^L = \{R_1^l \cup \dots \cup R_t^l\}_{l=1}^L$  for future training.

During training task  $t$ , gradients  $g_t^l$  are layer-wise constrained to be orthogonal to  $U_{t-1}^l$ . Particularly, assuming  $\mathbf{B}_{t-1}^l = [u_{t-1,1}^l, \dots, u_{t-1,N}^l]$  as the total  $N$  orthogonal basis of  $U_{t-1}^l$ , GPM (Saha et al., 2021) modifies the gradients as:

$$g_t^l = g_t^l - \text{Proj}_{U_{t-1}^l}(g_t^l) = g_t^l - \mathbf{B}_{t-1}^l (\mathbf{B}_{t-1}^l)^T g_t^l \quad (1)$$

In spite of alleviating forgetting, the strict orthogonal constraint on parameters hinders the forward knowledge transfer by the limited optimization space, thus compromising the performance of new tasks. TRGP (Lin et al., 2022) tackles this problem by selecting old tasks relevant to the current task and expanding corresponding frozen spaces as the trust regions. The scaled weight projection is further introduced for memory-efficient updating and storing the parameters by scaling the

basis. Considering that task  $i$  is selected as the trust region, the scaled weight projection is:

$$\text{Proj}_{U_i^l}^{S_i^l}(g_i^l) = \mathbf{B}_i^l \mathbf{S}_i^l (\mathbf{B}_i^l)^T g_i^l \quad (2)$$

where  $S_i^l$  denotes the scale matrix. For each task  $t$ , TRGP selects several different tasks as the trust regions  $U_i^l$ . During the training phase, gradient  $g_t^l$  is modified as:

$$g_t^l = g_t^l - \text{Proj}_{U_{t-1}^l}(g_t^l) + \sum_i \text{Proj}_{U_i^l}^{S_i^l}(g_t^l) \quad (3)$$

The parameters in the trust regions are retrained and the learned scale matrices are stored in the memory. During the inference phase, TRGP retrieves the model training on each task by replacing the parameters with the scale matrices. The parameters  $W_{t,I}^l$  used for inference on task  $t$  are:

$$W_{t,I}^l = W^l - \sum_i \text{Proj}_{U_i^l}(W^l) + \sum_i \text{Proj}_{U_i^l}^{S_i^l}(W^l) \quad (4)$$

However, as tasks come, storing scaling matrices introduces increasing extra parameters. Our experiments demonstrate that TRGP requires around 5000% amount of the parameters regarding the initial network after 20 tasks on MiniImageNet, see Fig. 3-(b). Therefore, we propose to facilitate forward knowledge transfer within a fixed network capacity by a restricted orthogonal constraint, allowing parameters updated in the direction oblique to the whole frozen space.

### 3.2. Overview

In this section, we introduce our Restricted Orthogonal Gradient prOjection (ROGO) framework. To better characterize the relationship between the gradient direction and the frozen space, we first define the angle between a given space and a vector in Definition 3.1.

---

#### Algorithm 1: Restricted orthogonal gradient projection

---

- 1: Initiate the frozen spaces  $U_0 = \{U_{i=1}^l\}$  as  $\emptyset$ s and optimize  $W_1$  for task 1.
  - 2: Compute frozen space  $U_1$  with representation matrices.
  - 3: **for**  $t \in 2, \dots, T$  **do**
  - 4: Initiate the relaxing space  $\{V_t^l\}_{i=1}^L$  as  $\emptyset$ s.
  - 5: **repeat**
  - 6: Fine-tune for predefined  $e_t$  epochs.
  - 7: Determine additional relaxing space  $\{V_t^{l,new}\}_{i=1}^L$  by the searching strategy in Section 3.3.
  - 8:  $V_t^l \leftarrow V_t^l \cup V_t^{l,new}$ .
  - 9: **until**  $V_t^{l,new}$  is  $\emptyset$ .
  - 10: Optimize  $\{W_t^l\}_{i=1}^L$  orthogonal to  $U_{t-1}^l \setminus V_t^l$  with the regularization terms in Section 3.4.
  - 11: Determine the representation space  $\{R_t^l\}_{i=1}^L$ .
  - 12: Update the frozen space by  $U_t^l = U_{t-1}^l \cup R_t^l$ .
  - 13: **end for**
- 

**Definition 3.1** (*Angle Between Vector and Space*). We denote the angle between two inputs as  $\Theta(\cdot)$  and the inner product between two vectors as  $\langle \cdot, \cdot \rangle$ . The angle between a vector  $v \in \mathbb{R}^n$  and a space  $U^{n \times c} \subset \mathbb{R}^n$  is defined as the minimum angle between the given vector  $v$  and any unit vector  $u \in U$ :

$$\Theta(v, U) = \arccos \max_{u \in U} \frac{\langle v, u \rangle}{\|v\|} \quad (5)$$

Instead of  $90^\circ$  in traditional gradient projection methods, we aim to relax the angle between the frozen space and the modified gradient to be an acute angle. However, directly rotating a vector in a high-dimensional space is too flexible to control. Hence, we notice Proposition 3.2.

**Proposition 3.2.** Given the full space  $R$  and a random space  $U \subset R$ , for any vector  $v \in U^\perp$ , where  $U^\perp = R \setminus U$  denotes the orthogonal complement of  $U$ , it can be modified to be oblique to the given space  $U$  in any angle, by scaling and combining a vector  $v'$  within a selected subspace  $U' \subseteq U$ .

According to Proposition 3.2, we can manipulate the gradient direction to be oblique to the entire frozen space  $U_{t-1}^l$  in any target angle by modifying the parameters within an appropriate subspace of  $U_{t-1}^l$ . Therefore, for each task  $t$ , instead of directly rotating the gradient, we introduce the layer-wise relaxing space  $V_t^l \subseteq U_{t-1}^l$ . During the training phase, we modify the gradients as:

$$g_t^l = g_t^l - \text{Proj}_{U_{t-1}^l \setminus V_t^l}(g_t^l) \quad (6)$$

By jointly optimizing the parameters within the selected relaxing space  $V_t^l$  and the original optimization space, the strict orthogonal constraint is relaxed. Hence, with a relaxing space whose basis is an empty set, namely  $\dim(V_t^l) = 0$ , our framework works exactly the same as GPM, under the orthogonal constraint. Therefore, our framework can be viewed as a generalization of GPM by regulating the relaxing space  $V_t^l$ . The relaxing space selection thus becomes the key procedure.

In Section 3.3, we introduce our searching strategy for the relaxing space. Given the relaxing space  $V_t^l$ , we involve complementary regularization loss to better consolidate previous knowledge. The procedure is provided in Section 3.4. For better illustration, we provide the procedure of our method in Algorithm 1. Besides, we propose ROGO-Exp in Section 3.5, involving additional parameters as expansion-based methods, to further validate the flexibility and effectiveness of our framework.

### 3.3. Searching the relaxing space

To determine the relaxing space  $V_t^l$ , we first construct the representation space  $R_{g,t}^l$  by the top- $k$  principal eigenvectors of the representative matrix  $[g_{t,1}^l, \dots, g_{t,N_t}^l]$ , where  $g_{t,j}^l$  denotes the gradient of the  $j$ th input. The estimated importance is further characterized by the angle from  $R_{g,t}^l$ . Here we define that a vector  $d$  is *relaxable* when:

$$\Theta(d, R_{g,t}^l) \leq \gamma_t^l \quad (7)$$

where  $\gamma_t^l$  is a predefined threshold. For task  $t$ , we aim to find the relaxing space  $V_t^l \subseteq U_{t-1}^l$  spanned all by *relaxable* vectors in  $U_{t-1}^l$ , namely:

$$\begin{cases} \max_{u \in V_t^l} \Theta(u, R_{g,t}^l) \leq \gamma_t^l \\ \min_{v \in U_{t-1}^{l,c}} \Theta(v, R_{g,t}^l) > \gamma_t^l \end{cases} \quad (8)$$

where  $U_{t-1}^{l,c} = U_{t-1}^l \setminus V_t^l$  denotes the complemented subspace of  $V_t^l$  with respect to  $U_{t-1}^l$ . Criterion (8) guarantees that all vectors in  $V_t^l$  are *relaxable* and there remains none *relaxable* vector in  $U_{t-1}^{l,c}$ . However, it is hard to construct the appropriate  $V_t^l$  directly from the high-dimensional  $U_{t-1}^l$ .

Therefore, we propose a simple yet efficient strategy to find  $V_t^l$ . Initiating  $V_t^l$  as  $\emptyset$ , we iteratively select the closest vector to  $R_{g,t}^l$  within  $U_{t-1}^{l,c}$  by  $\arg \min \Theta(d, R_{g,t}^l)$  and then append it into  $V_t^l$  as the basis if it is *relaxable*. This procedure is repeated until no *relaxable* vector is left, thus obtaining the target  $V_t^l$  consisting of all *relaxable* vectors within  $U_{t-1}^l$ .

To further substantiate the effectiveness of our searching strategy, here we introduce Theorem 3.3. According to Theorem 3.3, our strategy guarantees to find the maximum space within the whole solution set satisfying criterion (8). The pseudo-code of our searching strategy and detailed proof are provided in Appendices D and A.2.

**Theorem 3.3.** Denote  $S = \{u | \Theta(u, R_{g,t}^l) \leq \gamma_t^l \text{ and } u \in U_{t-1}^l\}$  as the whole solution set, the obtained relaxing space  $V_t^l$  guarantees  $\dim(V_t^l) \geq \dim(V)$ , for  $\forall V \subset S$ .

Moreover, we provide theoretical analysis on the upper bound of the dimension of  $V_t^l$ , which is also the number of iterations, to investigate the efficiency of our strategy. Here we introduce [Theorem 3.4](#), which guarantees that the dimension of  $V_t^l$  is no more than of the representation space  $R_{g,t}^l$ . In other words, the number of iterations is bounded. Hence, as  $R_{g,t}^l$  is constructed by the top- $k$  principal eigenvectors, the dimension of  $V_t^l$  is further regulated by hyperparameters, of which the ablation study is provided in [Section 5](#). We also include the detailed proof in [Appendix A.3](#).

**Theorem 3.4.** Denote  $k_r$  as the dimension of the representation space  $R_{g,t}^l$  and  $k_v$  as the dimension of the relaxing space  $V_t^l$ ,  $k_v \leq k_r$ , regardless of the frozen space  $U_{t-1}^l$ .

### 3.4. Constrained update in the relaxing space

Given the selected relaxing space  $V_t^l$ , the optimization space is enlarged, thus facilitating forward knowledge transfer. In the meanwhile, we also want to consolidate previous knowledge stored within  $V_t^l$ . One direct way is to incorporate additional regularization terms such as EWC ([Kirkpatrick et al., 2017](#)). However, regularization terms are designed for explicit parameters, which are not applicable to implicit subspace in our framework. Therefore, we borrow the scaled weight projection ([Lin et al., 2022](#)) to modify explicit parameters instead. With the scale matrix  $S_t^l$ , the gradient  $g_t^l$  is modified as:

$$g_t^l = g_t^l - \text{Proj}_{U_{t-1}^l}(g_t^l) + \text{Proj}_{V_t^l}^{S_t^l}(g_t^l) \quad (9)$$

Notice that we update the parameters within  $V_t^l$  with  $S_t^l$  after training each task, instead of storing the parameters for inference as TRGP. Parameters within  $V_t^l$  are then restrictedly fine-tuned by adding regularization terms on  $S_t^l$  of instead of directly on parameters. Specifically, the objective function is:

$$\mathcal{L}_t = \mathcal{L}(\mathcal{W}_t, D^{(t)}) + \sum_{l=1}^L \beta_l \|S_t^l - \mathbb{1}(S_t^l)\|_2^2 \quad (10)$$

where  $\mathbb{1}(\cdot)$  denotes the identity matrix with the same shape as the input matrix and  $\beta_l$  is the weight of the regularization term for layer  $l$ . During backpropagation, gradients within the relaxing space  $V_t^l$  are regulated by  $S_t^l$ . To further validate our strategy, we also equip TRGP with the above regularization terms for better comparison in [Section 4.3](#). Generally, in our method, we adopt our searching strategy to determine the relaxing space and modify those parameters with constraints on the scaling matrices.

However, during training, the direction of gradients shifts sharply and frequently due to the steep learning scope of deep neural networks. Diverse subspaces would be selected in different training phases. Therefore, we iteratively examine whether there remains *relaxable* vectors within the remaining frozen space  $U_{t-1}^{l,c}$  after limited epochs and then search for additional relaxing space  $V_t^{l,new}$ . If additional relaxing space  $V_t^{l,new}$  is involved, we expand the scaling matrix with identity matrices to accommodate the increased space. In contrast, TRGP maintains a fixed-size scaling matrix throughout training. As the  $U_{t-1}^l$  is fixed for each task, the number of iterations of our strategy is naturally limited. In the implementation, we further constrain the maximum number of iterations for efficiency.

In general, by iteratively searching for the relaxing space and modifying those gradients with additional regularization loss, our restricted orthogonal method optimizes the parameters in the direction oblique to the whole frozen space, thus facilitating forward knowledge transfer while consolidating previous knowledge within a fixed network capacity.

### 3.5. Extensions

To further validate our framework, we propose ROGO-Exp, a modified version of our proposed ROGO, directly storing the parameters within the relaxing space. Similar to TRGP, for each task  $t$ , we retrieve the corresponding relaxing space  $\{V_t^l\}_{l=1}^L$  and the scale matrices  $\{S_t^l\}_{l=1}^L$  during the inference phase. The modified parameters  $W_{t,l}^l$  used for inference on task  $t$  are:

$$W_{t,l}^l = W^l - \text{Proj}_{V_t^l}(W^l) + \text{Proj}_{V_t^l}^{S_t^l}(W^l) \quad (11)$$

where  $W^l$  denotes the parameters of layer  $l$  of current network. By replacing the parameters in the relaxing space with the parameters optimized in task  $t$ , the model achieves better performance, at the price of expansive computational cost in long task sequences. Further experimental results validate the efficiency of our ROGO-Exp against state-of-the-art expansion-based methods.

## 4. Experiments

### 4.1. Experimental setup

**Datasets:** Following [Saha et al. \(2021\)](#), we evaluate our approach on CIFAR-100 Split ([Krizhevsky and Hinton, 2009](#)), MiniImageNet ([Vinyals et al., 2016](#)), Permuted MNIST (PMNIST) ([Kirkpatrick et al., 2017](#)). Moreover, we introduce the Mixture benchmark, proposed by [Serra et al. \(2018\)](#), including eight datasets. In this work, we conduct experiments on Mixture with the seven tasks sequence except for TrafficSigns.<sup>1</sup> Details and statistics of the datasets can be found in [Appendix B.1](#). Moreover, we include the details of network architectures in [Appendix B.2](#).

**Baselines:** We compare ROGO with competitive and well-established methods within a fixed network capacity. For regularization-based methods, we compare against EWC ([Kirkpatrick et al., 2017](#)) and HAT ([Serra et al., 2018](#)). For gradient projection methods, we consider OGD ([Farajtabar et al., 2020](#)) and GPM ([Saha et al., 2021](#)). For replay-based methods, we adopt ER\_Res ([Chaudhry et al., 2019](#)) and A-GEM ([Chaudhry et al., 2018](#)). The memory buffer size for PMNIST, CIFAR-100 Split, MiniImageNet, and Mixture are 1000, 2000, 500, and 3000, respectively. Moreover, we consider TRGP ([Lin et al., 2022](#)) as a competitive approach under an expansion setting. Implementation details are listed in [Appendix B.3](#).

**Metrics:** Denote  $A_{i,j}$  as the test accuracy of task  $j$  after learning task  $i$  and  $b_i$  as the test accuracy of task  $i$  at random initialization. We employ the following three evaluation metrics: **(1) Average Accuracy (ACC)** ([Mirzadeh et al., 2020](#)): the average test accuracy evaluated after learning all tasks, defined as  $\frac{1}{T} \sum_{i=1}^T A_{T,i}$ ; **(2) Backward Transfer (BWT)** ([Lopez-Paz and Ranzato, 2017](#)): the average accuracy decrease after learning all tasks, defined as  $\frac{1}{T-1} \sum_{i=1}^{T-1} (A_{T,i} - A_{i,i})$ ; **(3) Forward Transfer ( $\Omega_{new}$ )** ([Kemker et al., 2018](#)): the average accuracy of new tasks, defined as  $\frac{1}{T-1} \sum_{i=2}^T (A_{i,i} - b_i)$ . As  $b_i$  stays still across different approaches, we consider  $\frac{1}{T-1} \sum_{i=2}^T A_{i,i}$ . Besides, FWT ([Lopez-Paz and Ranzato, 2017](#)) reflects the influence of the observed tasks on new tasks in a zero-shot manner, defined as  $\frac{1}{T-1} \sum_{i=2}^T (A_{i-1,i} - b_i)$ . In this paper, we mainly focus on  $\Omega_{new}$ , while FWT results are also provided. Detailed definitions are provided in [Appendix B.4](#).

<sup>1</sup> We fail to obtain TrafficSigns, as all the links provided in [Stallkamp et al. \(2011\)](#), [Serra et al. \(2018\)](#) and [Saha et al. \(2021\)](#) are expired.

**Table 1**

Comparison of average accuracy and forward transfer. For each task, we mark the best and the second best performance in **bold** and underline respectively. All results reported are averaged over 5 runs.

Method	CIFAR-100 Split		MiniImageNet	
	ACC (%)	$\Omega_{new}$ (%)	ACC (%)	$\Omega_{new}$ (%)
Multitask <sup>a</sup>	79.58 ± 0.54	–	69.46 ± 0.62	–
A-GEM <sup>b</sup>	63.98 ± 1.22	77.48 ± 0.40	57.24 ± 0.72	67.55 ± 1.20
ER_Res <sup>b</sup>	71.73 ± 0.63	77.13 ± 0.18	58.94 ± 0.85	69.48 ± 0.42
EWC	68.80 ± 0.88	69.87 ± 1.09	52.01 ± 2.53	<u>63.45 ± 2.80</u>
HAT	72.06 ± 0.50	71.50 ± 0.62	59.78 ± 0.57	62.63 ± 0.63
OGD	70.96 ± 0.32	<u>73.86 ± 0.57</u>	59.83 ± 0.62	62.02 ± 1.24
GPM	72.48 ± 0.40	71.53 ± 0.54	60.41 ± 0.61	61.63 ± 0.60
ROGO	<b>74.04 ± 0.35</b>	<b>75.22 ± 0.36</b>	<b>63.66 ± 1.24</b>	<b>63.81 ± 0.39</b>
Method	PMNIST		Mixture	
	ACC (%)	$\Omega_{new}$ (%)	ACC (%)	$\Omega_{new}$ (%)
Multitask <sup>a</sup>	96.70 ± 0.02	–	81.29 ± 0.23	–
A-GEM <sup>b</sup>	83.56 ± 0.16	97.42 ± 0.03	59.86 ± 1.01	85.87 ± 0.42
ER_Res <sup>b</sup>	87.24 ± 0.53	97.37 ± 0.05	75.07 ± 0.55	86.35 ± 0.22
EWC	89.97 ± 0.57	93.13 ± 0.70	69.62 ± 2.69	74.41 ± 1.00
HAT	–	–	77.54 ± 0.18	79.26 ± 0.15
OGD	82.56 ± 0.66	<b>97.40 ± 0.06</b>	74.38 ± 0.27	81.97 ± 0.53
GPM	93.91 ± 0.16	96.56 ± 0.04	77.49 ± 0.68	82.13 ± 0.37
ROGO	<b>94.20 ± 0.11</b>	96.26 ± 0.10	<b>77.91 ± 0.45</b>	<b>82.21 ± 0.42</b>

<sup>a</sup>Denotes under the non-incremental setting.

<sup>b</sup>Denotes requiring an extra data buffer.

## 4.2. Main results

We show the comparative results on four benchmarks in Table 1. The accuracy results of major baselines are adopted from GPM (Saha et al., 2021) and we implement all the baselines to get the forward knowledge transfer and forgetting results. Implementation details are provided in Appendix B.3. We run each experiment five times and report the mean results and the standard deviation. Detailed results including the forgetting are provided in Appendix C.2.

According to Table 1, ROGO dominates EWC across all benchmarks and generally outperforms HAT. Although HAT obtains comparable accuracy on Mixture, ROGO gains 2.7% better  $\Omega_{new}$  on average. For gradient projection methods, compared with GPM, ROGO achieves over 1% and 3% higher ACC on CIFAR-100 Split and MiniImageNet respectively while improving the forward knowledge transfer. We observe that OGD+ achieves the better  $\Omega_{new}$  on several datasets. Whereas, it fails on the accuracy with significant forgetting. Also, we notice that A-GEM and ER\_Res gain better  $\Omega_{new}$ , in spite of the inferior accuracy compared with GPM and especially ROGO. We assume that this is because replay-based methods explore the full optimization space with extra data, while both regularization-based and gradient projection methods constrain the optimization space to mitigate forgetting. In general, ROGO obtains the best accuracy and improves forward knowledge transfer without extra data buffers across all datasets.

Moreover, we compare the average accuracy after learning each task on CIFAR-100 Split with GPM, as GPM achieves the highest accuracy among selected baselines. As shown in Fig. 2-(a), ROGO increasingly gains superior performance by a larger margin, as the optimization space of GPM is gradually constrained by accumulated frozen spaces. We provide detailed results on other benchmarks in Appendix C.1. For better illustration, we also exhibit the accuracy evolution of specific tasks during sequential training in Fig. 2-(b). Here we choose the second task on CIFAR-100 Split. According to the results, ROGO universally outperforms GPM over the task sequence. Results of three randomly selected tasks are provided in Appendix C.4 respectively. In addition, we observe the accuracy on new tasks in Fig. 2-(c). Similar to the average accuracy, ROGO achieves increasingly better  $\Omega_{new}$  by

**Table 2**

Comparison of average accuracy and forward transfer with TRGP under an expansion setting. The percentages indicate the ratios of the rank of the relaxing space with respect to the frozen space. For each dataset, we mark the best and second best performance in **bold** and underline. Detailed results are provided in Appendix C.7.

Method	CIFAR-100 Split		MiniImageNet	
	ACC (%)	$\Omega_{new}$ (%)	ACC (%)	$\Omega_{new}$ (%)
TRGP	74.46 ± 0.22	75.01 ± 0.22	61.78 ± 1.94	<b>63.08 ± 1.40</b>
ROGO-Exp 50%	74.90 ± 0.37	<u>75.68 ± 0.43</u>	<b>63.46 ± 0.85</b>	<u>62.76 ± 0.64</u>
ROGO-Exp 80%	<u>74.97 ± 0.30</u>	75.46 ± 0.17	62.57 ± 1.32	62.32 ± 1.19
ROGO-Exp T%	<b>75.34 ± 0.61</b>	<b>75.86 ± 0.42</b>	<u>62.78 ± 1.00</u>	62.42 ± 1.28
Method	PMNIST		Mixture	
	ACC (%)	$\Omega_{new}$ (%)	ACC (%)	$\Omega_{new}$ (%)
TRGP	96.34 ± 0.11	97.07 ± 0.14	83.54 ± 1.15	<b>84.88 ± 0.95</b>
ROGO-Exp 50%	96.44 ± 0.20	97.07 ± 0.10	82.45 ± 0.49	84.13 ± 0.25
ROGO-Exp 80%	<u>96.76 ± 0.15</u>	<u>97.20 ± 0.08</u>	83.33 ± 0.31	84.60 ± 0.20
ROGO-Exp T%	<b>97.01 ± 0.08</b>	<b>97.26 ± 0.05</b>	<b>83.62 ± 0.26</b>	84.44 ± 0.42

**Table 3**

Comparison of average accuracy and forward transfer with TRGP within a fixed network capacity. We modify TRGP as TRGP-Reg with the same regularization terms.  $\beta$  indicates the regularization weight. For each dataset, we mark the best and second best performance in **bold** and underline. Detailed results are provided in Appendix C.7.

Method	CIFAR-100 Split		MiniImageNet	
	ACC (%)	$\Omega_{new}$ (%)	ACC (%)	$\Omega_{new}$ (%)
TRGP-Reg $\beta = 1$	72.01 ± 0.30	<u>73.17 ± 0.24</u>	55.81 ± 2.43	59.14 ± 1.00
TRGP-Reg $\beta = 5$	71.96 ± 0.40	<u>72.64 ± 0.67</u>	<b>58.81 ± 2.24</b>	<u>62.05 ± 0.74</u>
TRGP-Reg $\beta = 50$	<u>72.49 ± 0.09</u>	72.67 ± 0.26	22.69 ± 0.39	20.39 ± 0.36
ROGO	<b>74.04 ± 0.35</b>	<b>75.22 ± 0.36</b>	<b>63.66 ± 1.24</b>	<b>63.81 ± 0.39</b>
Method	PMNIST		Mixture	
	ACC (%)	$\Omega_{new}$ (%)	ACC (%)	$\Omega_{new}$ (%)
TRGP-Reg $\beta = 1$	<u>76.57 ± 2.69</u>	94.95 ± 0.23	73.31 ± 1.05	<b>83.64 ± 0.38</b>
TRGP-Reg $\beta = 5$	75.50 ± 3.96	95.47 ± 0.30	74.71 ± 0.85	<u>83.27 ± 0.24</u>
TRGP-Reg $\beta = 50$	76.56 ± 3.83	<u>95.88 ± 0.19</u>	17.36 ± 0.86	7.27 ± 0.98
ROGO	<b>94.20 ± 0.11</b>	<b>96.26 ± 0.10</b>	<b>77.91 ± 0.45</b>	82.21 ± 0.42

relaxing the strict orthogonal constraint. Results on other benchmarks included in Appendix C.3 further substantiate this phenomenon.

To understand our relaxing strategy better, we further conduct experiments on different thresholds  $\epsilon$ , which regulate the dimension of the frozen space. Saha et al. (2021) argue that  $\epsilon$  mediates the stability-plasticity dilemma by controlling the frozen space and thus is critical for GPM. However, ROGO enables the frozen space to be adaptively relaxed regarding the current task. Therefore,  $\epsilon$  plays a much less important role in ROGO. We present the performance of different  $\epsilon$  on CIFAR-100 Split in Fig. 2-(d). As shown in Fig. 2-(d), the performance of GPM drops significantly when  $\epsilon \geq 0.97$ , the optimal value reported in GPM, while ROGO consistently performs well even with  $\epsilon = 0.98$ . Generally, ROGO is more robust on the threshold  $\epsilon$ .

In brief, our approach universally outperforms selected baselines without extra data buffers. Achieving better average accuracy, ROGO also improves forward knowledge transfer by exploring a larger optimization space than GPM. To validate the efficiency of our relaxing strategy, we further compare ROGO-Exp with well-established and competitive expansion-based methods in the next section.

## 4.3. Comparison with expansion-based methods

The above experiments exhibit the superiority of our approach when maintaining a fixed network capacity. However, under the scenario with no limit on the network capacity, expansion-based methods achieve great performance by allocating new neurons or modules. Therefore, to further validate our strategy, we compare ROGO and ROGO-Exp with relative expansion-based methods.

In this section, we adopt TRGP (Lin et al., 2022), which expands the optimization space by retraining parameters within the selected

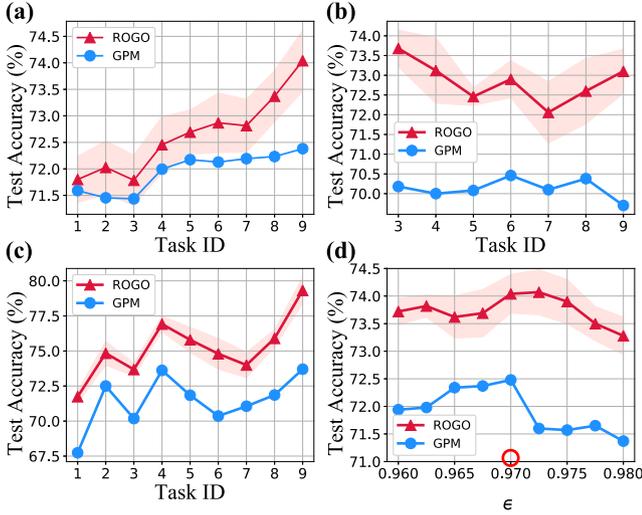


Fig. 2. Results on CIFAR-100 Split setting: (a) average accuracy after learning each task; (b) accuracy evolution of a randomly selected task; (c) accuracy tested on task  $i$  after learning task  $i$ ; (d) average accuracy of different  $\epsilon$ . The optimum  $\epsilon$  value of GPM is annotated by a red circle “o”, and the shaded area indicates the standard deviation.

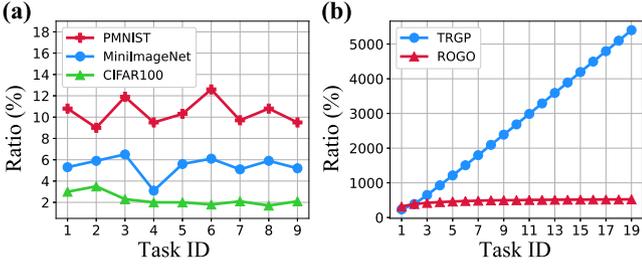


Fig. 3. (a) Relaxing ratios of the last layer on three datasets. (b) Ratios of the number of extra parameters concerning the number of the parameters of the network architecture on MiniImageNet.

trust regions, achieving superior performance. In the inference phase, TRGP reuses the parameters in corresponding trust regions memorized after learning this task. In contrast, GPM and ROGO only store the representation of the frozen space. Therefore, although indeed a stable network capacity is allocated for each task, the entire memory size of TRGP grows continually. As shown in Fig. 3-(b), after learning the last task on MiniImageNet, TRGP requires around 5000% extra parameters with respect to the network capacity. Results on other benchmarks further substantiate that TRGP introduces a significant number of extra parameters. Thus, we categorize TRGP as an expansion-based method here.

In this setting, the main difference between ROGO-Exp and TRGP is the strategy of deciding which part of the frozen space to reuse. We conduct experiments on all four benchmarks and report the results in Table 2. The percentages indicate the ratios of the rank of the relaxing space with respect to the corresponding frozen space. We evaluate two constant ratios and further use the ratios in TRGP, denoted as T%. As TRGP selects the top 2 tasks as the trust regions, T% is larger than 80% at most times. According to Table 2, ROGO-Exp universally outperforms TRGP, even by relaxing only 50% of the frozen space. Moreover, we observe that ROGO gains superior performance over both TRGP and ROGO-Exp on MiniImageNet. We assume that the reason is the positive backward transfer. Training the network on new tasks also benefits the previous task. In general, our approach achieves better performance with a comparable size of the relaxing space, which substantiates the efficiency of our searching strategy.

Table 4

Ablation study of  $\zeta = \cos \gamma$  on CIFAR100-Split, where  $\gamma$  is the threshold for the relaxing strategy.  $\zeta_{conv}$  and  $\zeta_{fc}$  denote the threshold for convolutional and fully connected layers, respectively. Here we report the relaxing ratio of each layer.

$\zeta_{conv}$	$\zeta_{fc}$	Conv1	Conv2	Conv3	Fc1	Fc2	ACC	BWT
0.20	0.20	38.09%	34.63%	51.11%	80.75%	60.56%	71.97%	-2.97%
0.50	0.50	30.26%	28.75%	40.53%	34.78%	11.26%	72.58%	-2.41%
0.80	0.80	28.51%	18.71%	25.68%	5.94%	0.23%	73.34%	-2.04%
0.90	0.90	27.87%	14.16%	18.65%	1.08%	0.07%	73.68%	-1.76%
<b>0.95</b>	<b>0.90</b>	26.31%	10.51%	13.63%	0.97%	0.04%	<b>74.04%</b>	-1.43%
0.95	0.95	24.12%	10.52%	13.83%	0.08%	0.00%	73.87%	-1.53%
0.99	0.99	18.66%	5.51%	9.51%	0.00%	0.00%	73.70%	-1.34%

Table 5

Ablation study of  $\beta$  on CIFAR100-Split, where  $\beta$  is the regularization weight.

$\beta$	ACC	BWT
0.0	72.43%	-3.03%
0.1	72.94%	-2.57%
0.5	73.68%	-1.88%
<b>1.0</b>	<b>74.04%</b>	-1.34%
5.0	73.77%	-1.42%
10.0	73.42%	-0.87%
50.0	72.65%	-0.83%

We further modify TRGP as TRGP-Reg with similar regularization terms on the scale matrices as our ROGO to compare the relaxing strategies. We report the results on four benchmarks with three representative regularization weights  $\beta$  on TRGP-Reg in Table 3. As shown in Table 3, ROGO significantly outperforms TRGP-Reg, especially on PMNIST, gaining around 20% ACC improvement. Generally, ROGO achieves better or comparable  $\Omega_{new}$  than TRGP under or without the constraint of a fixed network capacity. Detailed results in this section are provided in Appendix C.7.

## 5. Analysis and discussion

To gain a deeper insight into ROGO, we investigate the trend of scales of the space relaxed by our strategy. With the theoretical upper bound of the rank of the relaxed subspace provided in Theorem 3.4, we further inspect the ratios of the relaxing spaces concerning corresponding frozen spaces in practice. Fig. 3-(a) exhibits the results of the last layer on three datasets. Generally, relaxing ratios maintain a stable trend, fluctuating smoothly within a small range over sequential tasks. As different tasks explore different optimization directions, ideal relaxing spaces vary across tasks, consistent with our fluctuation. To further investigate our method, we provide the ablation study in Tables 4 and 5.

**Ablation study of the relaxing ratios.** ROGO mediates the stability-plasticity dilemma by controlling the dimension and flexibility of the relaxing space by  $\gamma$  in Eq. (8). For better illustration, here we represent  $\gamma$  by  $\zeta = \cos \gamma$ . We first conduct an ablation study of  $\zeta$ , reporting both the performance and the relaxing ratios in Table 4, where  $\zeta_{conv}$  denotes the hyper-parameter for convolutional layers and  $\zeta_{fc}$  denotes the hyper-parameter for fully connected layers. As shown in Table 4, increasing  $\zeta$  gradually constrains the scale of the relaxing space, thus alleviating the forgetting. On the other hand, a small  $\zeta$  guarantees sufficient forward knowledge transfer, while fails on consolidating previous knowledge. Generally, our method persists in a superb performance with  $\zeta \geq 0.80$ .

**Ablation study of the regularization weights.** Moreover, we observe the performance of ROGO with various regularization weights  $\beta$  in Table 5. In this work, we adopt a consistent  $\beta$  for the entire network. According to Table 5, Similarly, we observe less forgetting on larger  $\beta$ , which constrains the update of parameters within the relaxing space more strictly. However, strict constraints also lead to limited performance on new tasks as discussed in Section 4. In a nutshell,  $\zeta$

and  $\beta$  operate together in ROGO to overcome catastrophic forgetting and enhance forward transfer.

**Training time.** The dimension of the frozen spaces keeps growing as the tasks accumulate, leading to expanding range for searching relaxing spaces. Therefore, the computation complexity and time consumption are supposed to increase gradually. We further reported the time consumption comparison on CIFAR-100 Split and MiniImageNet in [Appendix C.5](#), where ROGO takes around 50% more time than GPM, similar to TRGP. In general, the practical efficiency of our approach is acceptable.

## 6. Conclusion

In this paper, we proposed ROGO, a novel continual learning framework that facilitates forward knowledge transfer in gradient projection methods within a fixed network capacity. ROGO imposes a restricted orthogonal constraint allowing parameters to be updated in the direction oblique to the whole frozen space, which thus explores a larger optimization space. Extensive experiments demonstrate that our ROGO framework surpasses related state-of-the-art approaches on diverse benchmarks. Moreover, we proposed ROGO-Exp that allows network expansion with the relaxing space, which achieves better performance than related expansion-based methods.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61925601, 62276152). We thank all anonymous reviewers for their valuable comments and suggestions on this work.

## Appendix A. Proof

In this section, we sequentially provide the proof of [Theorems 3.3](#) and [3.4](#). For better comprehension, we first introduce complementary [Lemma A.1](#). For simplification, all vectors here are assumed to be unit vectors, namely  $\|v\| = 1$ .

**Lemma A.1.** Denote the relaxed space as  $V = \text{span}\{v_1, v_2, \dots, v_n\}$ , where  $v_n$  is the last base included in  $V$ . Given representation space  $U$ ,  $\forall v \in V$ , we have  $\theta(v, U) \leq \theta(v_n, U)$ .

### A.1. Proof of [Lemma A.1](#)

Depicting the angle by projection form, [Lemma A.1](#) can be expressed as:

$$\forall v \in V, \|\text{Proj}_U(v)\| \geq \|\text{Proj}_U(v_n)\| \quad (\text{A.1})$$

Denote  $\mathbf{B} = [u_1, \dots, u_m]$  as the representative matrix of the representation space  $U = \text{span}\{u_1, \dots, u_m\}$ , where  $u_i$  is the  $i$ th normalized orthogonal base of  $U$ . [Lemma A.1](#) can be further expressed as:

$$\forall v \in V, v^T \mathbf{B} \mathbf{B}^T v \geq v_n^T \mathbf{B} \mathbf{B}^T v_n \quad (\text{A.2})$$

As  $v_i$ s are the basis sequentially appended by our searching strategy, for any  $i \leq j$ , we have:

$$v_i^T \mathbf{B} \mathbf{B}^T v_i \geq v_j^T \mathbf{B} \mathbf{B}^T v_j \quad (\text{A.3})$$

Consider a special case where the current relaxed subspace has only one base, denoted by  $V = \text{span}\{v_1\}$ , namely  $n = 1$ . Obviously, [Lemma A.1](#) is true in this case. Thus, we consider the general case that  $n \geq 2$ .

We provide proof by inductive reasoning. Note that as we assume all vectors to be unit vectors, we have  $\sum_i \|w_i\|_2^2 = 1$ .

First, we consider a special case  $V = \text{span}\{v_1, v_2\}$ , namely  $n = 2$ . Assume there exists  $v = w_1 v_1 + w_2 v_2$  that  $v^T \mathbf{B} \mathbf{B}^T v < v_2^T \mathbf{B} \mathbf{B}^T v_2$ , with  $w_1^2 + w_2^2 = 1$ , we have:

$$w_1^2 v_2^T \mathbf{B} \mathbf{B}^T v_2 > w_1^2 v_1^T \mathbf{B} \mathbf{B}^T v_1 + 2w_1 w_2 v_1^T \mathbf{B} \mathbf{B}^T v_2 \quad (\text{A.4})$$

Construct  $v' = w_2 v_1 - w_1 v_2$ , we have:

$$\begin{aligned} v'^T \mathbf{B} \mathbf{B}^T v' &= w_2^2 v_1^T \mathbf{B} \mathbf{B}^T v_1 + w_1^2 v_2^T \mathbf{B} \mathbf{B}^T v_2 - 2w_1 w_2 v_1^T \mathbf{B} \mathbf{B}^T v_2 \\ &> w_2^2 v_1^T \mathbf{B} \mathbf{B}^T v_1 + w_1^2 v_2^T \mathbf{B} \mathbf{B}^T v_2 + w_1^2 v_1^T \mathbf{B} \mathbf{B}^T v_1 - w_1^2 v_2^T \mathbf{B} \mathbf{B}^T v_2 \\ &= v_1^T \mathbf{B} \mathbf{B}^T v_1 \end{aligned} \quad (\text{A.5})$$

which contradicts  $\theta(v_1, U) \leq \theta(v, U)$  for  $\forall v \in \text{span}\{v_1, v_2\}$ .

Then we consider the general case  $V = \text{span}\{v_1, \dots, v_i\}$ . For  $\forall v \in V$ , we have  $\theta(v, U) \leq \theta(v, U)$ . After  $v_{i+1}$  is included, we assume that there exists  $v \in V$  that  $\theta(v, U) > \theta(v_{i+1}, U)$ . Then we can find the maximum  $s$  satisfying that there exists  $v = \sum_{i=1}^s w_i v_i + w_{i+1} v_{i+1}$  that  $\theta(v, U) > \theta(v_{i+1}, U)$  and for  $\forall v' = \sum_{i=1}^{s-1} w_i v_i + w_{i+1} v_{i+1}$ , we have  $\theta(v', U) \leq \theta(v_{i+1}, U)$ . When  $s = 1$ , it is similar to the special case, so the proof is omitted. Thus, we consider the case where  $s \geq 2$ . For simplification, we express  $v$  as  $v = c_0 v_0 + c_1 v_s + c_2 v_{i+1}$  with  $v_0 = \sum_{i=1}^{s-1} a_i v_i$ , where  $c_i$ s and  $a_i$ s are coefficients. We have:

$$(c_0 v_0 + c_1 v_s + c_2 v_{i+1})^T \mathbf{B} \mathbf{B}^T (c_0 v_0 + c_1 v_s + c_2 v_{i+1}) < v_{i+1}^T \mathbf{B} \mathbf{B}^T v_{i+1} \quad (\text{A.6})$$

As  $\theta(w_1 v_0 + w_2 v_s, U) \leq \theta(v_s, U) \leq \theta(v_{i+1}, U)$ , we have:

$$(w_1 v_0 + w_2 v_s)^T \mathbf{B} \mathbf{B}^T (w_1 v_0 + w_2 v_s) \geq v_s^T \mathbf{B} \mathbf{B}^T v_s \quad (\text{A.7})$$

which is:

$$w_1^2 v_0^T \mathbf{B} \mathbf{B}^T v_0 + 2w_1 w_2 v_0^T \mathbf{B} \mathbf{B}^T v_s \geq w_1^2 v_s^T \mathbf{B} \mathbf{B}^T v_s \geq w_1^2 v_i^T \mathbf{B} \mathbf{B}^T v_i \quad (\text{A.8})$$

Similarly we have:

$$w_1^2 v_0^T \mathbf{B} \mathbf{B}^T v_0 + 2w_1 w_2 v_0^T \mathbf{B} \mathbf{B}^T v_i \geq w_1^2 v_i^T \mathbf{B} \mathbf{B}^T v_i \quad (\text{A.9})$$

Then we can express Eq. (A.6) as:

$$v_{i+1}^T \mathbf{B} \mathbf{B}^T v_{i+1} > (c_0^2 + c_2^2) v_i^T \mathbf{B} \mathbf{B}^T v_i + c_1^2 v_s^T \mathbf{B} \mathbf{B}^T v_s + 2c_1 c_2 v_s^T \mathbf{B} \mathbf{B}^T v_i \quad (\text{A.10})$$

As  $\|v\| = \|v_i\| = 1$ ,  $c_0^2 + c_1^2 + c_2^2 = 1$ . Then we have:

$$-2c_1 c_2 v_s^T \mathbf{B} \mathbf{B}^T v_i > c_1^2 v_s^T \mathbf{B} \mathbf{B}^T v_s - c_1^2 v_{i+1}^T \mathbf{B} \mathbf{B}^T v_{i+1} \quad (\text{A.11})$$

Construct  $v' = \frac{c_2 v_s - c_1 v_{i+1}}{\sqrt{c_1^2 + c_2^2}}$ , we have:

$$\begin{aligned} v'^T \mathbf{B} \mathbf{B}^T v' &= \frac{1}{c_1^2 + c_2^2} (c_2^2 v_s^T \mathbf{B} \mathbf{B}^T v_s + c_1^2 v_{i+1}^T \mathbf{B} \mathbf{B}^T v_{i+1} - 2c_1 c_2 v_s^T \mathbf{B} \mathbf{B}^T v_i) \\ &> \frac{1}{c_1^2 + c_2^2} (c_2^2 v_s^T \mathbf{B} \mathbf{B}^T v_s + c_1^2 v_{i+1}^T \mathbf{B} \mathbf{B}^T v_{i+1} \\ &\quad + c_1^2 v_s^T \mathbf{B} \mathbf{B}^T v_s - c_1^2 v_{i+1}^T \mathbf{B} \mathbf{B}^T v_{i+1}) \\ &= v_s^T \mathbf{B} \mathbf{B}^T v_s \end{aligned} \quad (\text{A.12})$$

which contradicts  $\theta(v_s, U) \leq \theta(v, U)$  for  $\forall v \in \text{span}\{v_s, v_{s+1}, \dots, v_i, v_{i+1}\}$ .

Thus, for  $\forall v \in V = \text{span}\{v_1, \dots, v_n\}$ , we have  $\theta(v, U) \leq \theta(v_n, U)$ .

### A.2. Proof of [Theorem 3.3](#)

Here we provide proof of [Theorem 3.4](#). Denote the whole solution set as  $S = \{u | \theta(u, U) \leq \gamma \text{ and } u \in U^f\}$  where  $U$  is the representation space,  $U^f$  is the frozen space and  $\gamma$  is the threshold. [Theorem 3.3](#) can be expressed as that all subspace  $V' \subseteq S$  satisfies  $\dim(V') \leq \dim(V)$ , where  $V$  is the relaxing space obtained by our searching strategy.

For the sake of contradiction, we assume there exists  $V' \subseteq S$  that  $\dim(V') > \dim(V)$ . Denote  $V_f^c$  as the orthogonal complement of  $V$  with respect to the frozen space  $U^f$ , as the relaxing space is a subspace of the frozen space. According to Lemma A.1, for  $\forall v \in V_f^c$ , we have  $\Theta(v, U) > \gamma$ . We also have:

$$\dim(V' \cap V_f^c) = \dim(V') + \dim(V_f^c) - \dim(V' + V_f^c) > 0 \quad (\text{A.13})$$

Thus, there exists  $v' \in V'$  that  $v' \in V_f^c$ , namely there exists  $v' \in S$  that  $\Theta(v', U) > \gamma$ , which is contradict. Therefore, the relaxing space obtained by our searching strategy takes up the maximum subspace of the whole solution set.

### A.3. Proof of Theorem 3.4

Here we provide proof of Theorem 3.4. Denote the relaxing space and the representation space as  $V$  and  $U = \text{span}\{u_1, \dots, u_{k_r}\}$  respectively. With Lemma A.1, we have  $\forall v \in V$ ,  $\Theta(v, U) \leq \Theta(v_{k_r}, U) < \frac{\pi}{2}$ , where  $k_r$  denotes  $\dim(V)$ , the dimension of  $V$ . In other words,

$$\forall v \in V, v \not\perp U \quad (\text{A.14})$$

Similar to Theorem 3.4, assume the dimension of  $V$  is larger than  $k_r$ , namely  $\dim(V) > \dim(U) = k_r$ . Denote  $U^c$  as the orthogonal complement of  $U$  with respect to the whole space  $\mathbb{R}$ . Obviously,  $\dim(U^c) = n - k_r$ , where  $n$  is the dimension of  $\mathbb{R}$ . Then we have:

$$\begin{aligned} \dim(V \cap U^c) &= \dim(V) + \dim(U^c) - \dim(V + U^c) \\ &> k_r + (n - k_r) - n = 0 \end{aligned} \quad (\text{A.15})$$

Thus, there exists  $v' \in V$  such that  $v' \in U^c$  too. As  $U^c$  is the orthogonal complement,  $\forall u \in U^c, u \perp U$ . Then we have  $v' \perp U$ , which contradicts Eq. (A.14). Therefore, the assumption is aborted. We have that  $k_v \leq k_r$ . The upper bound of the dimension of  $V$  is  $k_r$ , namely the dimension of the representation space  $U$ .

## Appendix B. Experimental setup

### B.1. Datasets

Here we introduce the datasets we use for evaluation. (1) **CIFAR-100 Split** Saha et al. (2021) constructed **CIFAR-100 Split**, by splitting CIFAR100 (Krizhevsky and Hinton, 2009) into 10 tasks where each task has 10 classes. (2) **MiniImageNet** Following Saha et al. (2021), we split MiniImageNet (Vinyals et al., 2016) into 20 sequential tasks with 5 classes each. (3) **Permuted MNIST (PMNIST)** PMNIST (Kirkpatrick et al., 2017) is a variant of MNIST (LeCun et al., 1998) where each task has a different permutation of inputting images, consists of 10 sequential tasks with 10 classes each. (4) **Mixture** Serra et al. (2018) first proposed Mixture consisting of 8 datasets, including CIFAR-10 (Krizhevsky and Hinton, 2009), MNIST (LeCun et al., 1998), CIFAR-100 (Krizhevsky and Hinton, 2009), SVHN (Netzer et al., 2011), FashionMNIST (Xiao et al., 2017), TrafficSigns (Stallkamp et al., 2011), FaceScrub (Ng and Winkler, 2014), and NotMNIST (Bulatov, 2011), from which Ebrahimi et al. (2020) further constructed 5-Datasets. Here we follow the original harder benchmark. Particularly, we consider all tasks as a sequence except TrafficSigns (Stallkamp et al., 2011), which we failed to access. (5) **CIFAR-100 Sup** In addition, following Yoon et al. (2019), we adopt **CIFAR-100 Sup** consisting of 20 superclasses as sequential tasks. Here we adopt the five different predefined orders proposed by Yoon et al. (2019) and report the main results in Appendix C.7. Among all evaluated datasets, PMNIST is a benchmark under the domain-incremental scenario, while other four datasets are under the task-incremental scenario.

Moreover, we provide the statistics of selected datasets in Tables B.6 and B.7. For the Mixture benchmark, the images of MNIST, FashionMNIST, and notMNIST are replicated across all RGB channels following Serra et al. (2018).

**Table B.6**

Statistics of CIFAR-100 Split, MiniImageNet, and PMNIST.

	CIFAR-100 Split	CIFAR-100 Sup	MiniImageNet	PMNIST
Image size	32 × 32	32 × 32	84 × 84	28 × 28
Channels	3	3	3	1
Classes	100	100	100	10
Tasks	10	20	20	10
Classes/task	10	5	5	10
Training samples/task	4750	2375	2375	54,000
Validation samples/task	250	125	125	6000
Testing samples/task	1000	500	500	10,000

### B.2. Model details

**MLP architecture:** We adopt a 3-layer model including two hidden layers with 100 neurons each for the PMNIST setting, the same as Lopez-Paz and Ranzato (2017). ReLU is used as the activate function here and for all other architectures. Also, we use softmax with cross entropy loss on all settings.

**AlexNet architecture:** For CIFAR-100 Split setting, we adopt the same network as Serra et al. (2018) with batch normalization, including two fully connected layers and three convolutional layers. The convolutional layers have 4 × 4, 3 × 3, and 2 × 2 kernel sizes with 64, 128, and 256 filters respectively. After each convolutional layer, we add batch normalization and 2 × 2 max-pooling. Each fully connected layer has 2048 units. For the first two layers, we use the dropout of 0.2, and for the rest layers, we use the dropout of 0.5.

**Modified LeNet-5 architecture:** For the CIFAR-100 Sup setting, a modified LeNet-5 architecture consisting of two convolutional layers and two fully connected layers is adopted, similar to Saha et al. (2021). Max-pooling of 3 × 2 is used after each convolutional layer. The last two layers have 800 and 500 units respectively.

**Reduced ResNet-18 architecture:** We adopt the same reduced ResNet-18 architecture as Saha et al. (2021) for the MiniImageNet and Mixture settings, using 2 × 2 average-pooling before the classifier layer instead of the 4 × 4 average-pooling used by Lopez-Paz and Ranzato (2017). Moreover, we present the dimension of the representation space of each layer of our architectures in Table B.8.

### B.3. Implementation details

We use the official implementation of GPM (Saha et al., 2021), HAT (Serra et al., 2018), and TRGP (Lin et al., 2022). We implement A-GEM and ER\_Res with the official implementation by Chaudhry et al. (2018) and implement EWC with the implementation by Serra et al. (2018). For OGD (Farajtabar et al., 2020), we use the implementation by Bennani et al. (2020). Following Saha et al. (2021) and Lin et al. (2022), we run all experiments five times on an established seed without fixing the cuda settings for a fair comparison. Particularly, we use five random seeds on PMNIST where there is no diversity on a single seed. For CIFAR-100 Sup, we use five different orders provided by Yoon et al. (2019). Following Saha et al. (2021), we report the experimental results of replay-base methods A-GEM and ER\_Res on the Mixture dataset with the same buffer size as GPM and ROGO, which is 8.98M in terms of the number of parameters for Resnet18 architecture.

On CIFAR-100 Split, MiniImageNet, and PMNIST, we follow the hyper-parameters utilized by Saha et al. (2021) and Lin et al. (2022), including learning rate, batch size, and the threshold  $\epsilon$ . On Mixture, as we adopt the same network architecture Saha et al. (2021) use on their 5-Dataset setting, we follow the provided learning rate and batch size as well.

As discussed in Section 5, the threshold  $\zeta = \cos \gamma$  controls the criterion of the relaxing space. For CIFAR100-Split and PMNIST, we use  $\zeta = 0.95$  for convolutional layers and  $\zeta = 0.9$  for fully connected layers. For MiniImageNet and Mixture, we use the same  $\zeta$  for all layers, 0.95 and 0.9 respectively. The regularization weight  $\beta$  is set as 5 for

**Table B.7**  
Statistics of mixture benchmark.

Dataset	Classes	# Training	# Validation	# Testing
CIFAR-10 (Krizhevsky and Hinton, 2009)	10	47,500	2500	10,000
MNIST (LeCun et al., 1998)	10	57,000	3000	10,000
CIFAR-100 (Krizhevsky and Hinton, 2009)	100	47,500	2500	10,000
SVHN (Netzer et al., 2011)	10	69,595	3662	26,032
FashionMNIST (Xiao et al., 2017)	10	57,000	3000	10,000
FaceScrub (Ng and Winkler, 2014)	100	19,570	1030	2,289
NotMNIST (Bulatov, 2011)	10	16,011	842	1,873

**Table B.8**  
Dimension of the representation space of each layer.

Network	Depth	Dimension of the representation space
MLP	3 layers	784; 100; 100
AlexNet	5 layers	48; 576; 512; 1024; 2048
LeNet-5	4 layers	75; 500; 3200; 800
ResNet-18	17 layers and 3 short-cut connections	27; 180; 180; 180; 180; 180; 360; 20; 360; 360; 360; 720; 40; 720; 720; 720; 1440; 80; 1440; 1440

the ResNet18 architecture and 1 for others. Moreover, we limit the max iteration times to 2 for CIFAR100-Split and MiniImageNet for efficiency. Particularly, we run all the experiments on a single NVIDIA GeForce RTX 2080 Ti GPU.

#### B.4. Metrics

Here we present the detailed definitions of the metrics evaluating the forward knowledge transfer.  $\Omega_{new}$  (Kemker et al., 2018). Denote  $b_i$  as the test accuracy of task  $i$  at random initialization, FWT, first proposed by Lopez-Paz and Ranzato (2017), is defined as  $\text{FWT} = \frac{1}{T-1} \sum_{i=2}^T (A_{i-1,i} - b_i)$ , evaluating the zero-shot performance of the initialization with respect to the observed tasks. While  $\Omega_{new}$ , first proposed by Kemker et al. (2018), is defined as  $\Omega_{new} = \frac{1}{T-1} \sum_{i=2}^T (A_{i,i} - b_i)$ , reflecting the test accuracy on new tasks based on the learnt knowledge. As  $b_i$  stays still across different approaches, we consider  $\Omega_{new} = \frac{1}{T-1} \sum_{i=2}^T A_{i,i}$  for simplicity. For this simplified  $\Omega_{new}$ , we have:  $\Omega_{new} = \frac{T}{T-1} \text{ACC} - \text{BWT} - \frac{1}{T-1} A_{1,1}$ , with the ACC and BWT defined in Section 4.1.

## Appendix C. Experimental results

### C.1. Final accuracy

We provide the test accuracy after learning each task on other benchmarks here. As discussed in Section 4, our ROGO universally outperforms GPM over the task sequence on all benchmarks (see Fig. C.4).

### C.2. Backward knowledge transfer

As mentioned in Section 4.1, final accuracy (ACC), forgetting (BWT) and forward knowledge transfer ( $\Omega_{new}$ ) are jointly considered to evaluate a continual learner. According to Table C.9, ROGO achieves the best forgetting on MiniImageNet and PMNIST datasets. Despite our relaxing strategy focusing on facilitating the forward knowledge transfer, ROGO gains better final accuracy with comparable forgetting over all benchmarks. Generally speaking, ROGO achieves superior performance than previous baselines with a fixed network capacity.

**Table C.9**

Comparison of average accuracy and forgetting tested after learning all tasks. *Multitask* is under the non-incremental setting. For each task, we mark the best and the second best performance in **bold** and underline respectively. All results reported are averaged over 5 runs.

Method	CIFAR-100 Split		MiniImageNet	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Multitask*	79.58 ± 0.54	–	69.46 ± 0.62	–
A-GEM†	63.98 ± 1.22	–16.30 ± 1.19	57.24 ± 0.72	–10.95 ± 1.29
ER_Res†	71.73 ± 0.63	–5.50 ± 0.76	58.94 ± 0.85	–8.84 ± 0.56
EWC	68.80 ± 0.88	–2.31 ± 0.76	52.01 ± 2.53	–12.14 ± 2.28
HAT	72.06 ± 0.50	<b>–0.12 ± 0.42</b>	59.78 ± 0.57	–3.31 ± 0.05
OGD	70.96 ± 0.32	<u>73.86 ± 0.57</u>	59.83 ± 0.62	62.02 ± 1.24
GPM	72.48 ± 0.40	<u>–0.72 ± 0.27</u>	60.41 ± 0.61	–1.54 ± 0.34
ROGO	<b>74.04 ± 0.35</b>	–1.43 ± 0.43	<b>63.66 ± 1.24</b>	<b>–0.09 ± 0.94</b>
Method	PMNIST		Mixture	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
Multitask*	96.70 ± 0.02	–	81.29 ± 0.23	–
A-GEM†	83.56 ± 0.16	–14.57 ± 2.33	59.86 ± 1.01	–29.37 ± 1.11
ER_Res†	87.24 ± 0.53	–10.24 ± 1.58	75.07 ± 0.55	–11.81 ± 0.82
EWC	89.97 ± 0.57	–3.58 ± 1.22	69.62 ± 2.69	–6.00 ± 4.09
HAT	–	–	77.54 ± 0.18	<b>–0.55 ± 0.28</b>
GPM	93.91 ± 0.16	<u>–3.30 ± 0.09</u>	77.49 ± 0.68	–4.70 ± 0.50
ROGO	<b>94.20 ± 0.11</b>	<b>–2.44 ± 0.13</b>	<b>77.91 ± 0.45</b>	<u>–4.55 ± 0.36</u>

**Table C.10**

The accuracy tested on task  $i$  after learning task  $i$  and  $\Omega_{new}$  on CIFAR-100 Split.

Method	1	2	3	4	5	6	7	8	9	Avg ( $\Omega_{new}$ )
GPM	67.7	72.5	70.1	73.6	71.8	70.3	71.0	71.8	73.7	71.5
ROGO	71.7	74.8	73.6	76.9	75.8	74.7	74.0	75.9	79.3	75.2

### C.3. Forward knowledge transfer

We provide the detailed forward knowledge transfer performance on all four benchmarks here. First, we present the results of  $\Omega_{new}$  and the detailed accuracy of each task after learning it in Tables C.10 to C.13. According to Tables C.12 and C.13, ROGO achieves a similar forward knowledge transfer compared with GPM. For other benchmarks, ROGO improves  $\Omega_{new}$  by 2.7% and 1.8% on CIFAR-100 Split and MiniImageNet respectively, as shown in Tables C.10 and C.11.

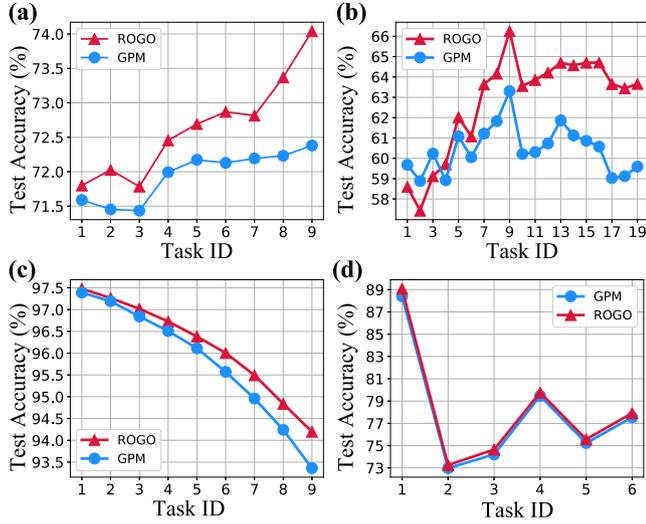
Moreover, we provide the result of FWT (using the definition in Lopez-Paz and Ranzato (2017)) on all benchmarks in Table C.14. According to Table C.14, although our method facilitates the forward knowledge transfer reflected by  $\Omega_{new}$ , ROGO achieves better FWT than GPM on all three task-incremental benchmarks, namely better zero-shot performance.

### C.4. Accuracy evolution

Here we present the accuracy tested on three randomly selected tasks immediately after learning them. We select the 2nd, 4th, and 6th tasks for all benchmarks. Generally, ROGO outperforms GPM on selected tasks over the sequence. We further notice that the improvement is more significant on later tasks owing to a larger relaxing space, as discussed in Section 5 (see Figs. C.5–C.7).

**Table C.11**The accuracy tested on task  $i$  after learning task  $i$  and  $\Omega_{new}$  on MiniImageNet.

Method	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Avg ( $\Omega_{new}$ )
GPM	59.0	57.1	65.6	59.6	76.2	57.2	66.5	72.8	81.5	42.0	59.6	62.1	62.3	58.9	57.2	59.0	50.7	65.7	57.3	61.6
ROGO	57.1	55.1	66.2	61.3	79.8	61.4	66.6	73.5	83.6	42.8	65.1	63.4	67.1	64.6	61.6	61.6	50.0	68.2	60.8	63.7

**Fig. C.4.** Average accuracy after learning each task on (a) CIFAR100-Split, (b) MiniImageNet, (c) PMNIST, and (d) Mixture.**Table C.12**The accuracy tested on task  $i$  after learning task  $i$  and  $\Omega_{new}$  on PMNIST.

Method	1	2	3	4	5	6	7	8	9	Avg ( $\Omega_{new}$ )
GPM	97.5	97.4	97.0	96.8	96.5	96.2	96.2	95.8	95.1	96.5
ROGO	97.4	97.2	97.0	96.5	96.3	96.1	95.7	94.9	94.8	96.2

**Table C.13**The accuracy tested on task  $i$  after learning task  $i$  and  $\Omega_{new}$  on Mixture.

Method	1	2	3	4	5	6	Avg ( $\Omega_{new}$ )
GPM	99.0	43.7	87.3	99.1	69.9	93.4	82.1
ROGO	99.1	42.9	87.6	99.1	70.6	93.5	82.2

**Table C.14**

Comparison of forward knowledge transfer between GPM and ROGO, evaluated by FWT.

Methods	CIFAR-100 Split	MiniImageNet	PMNIST	Mixture
GPM	$-0.65 \pm 0.18$	$-0.26 \pm 0.88$	$+0.66 \pm 1.17$	$-0.52 \pm 1.49$
ROGO	$+0.20 \pm 0.36$	$+0.30 \pm 0.29$	$-0.63 \pm 0.97$	$+0.13 \pm 0.77$

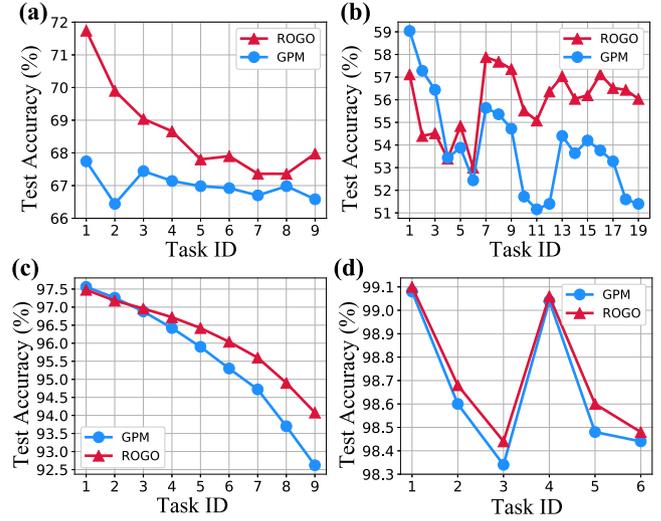
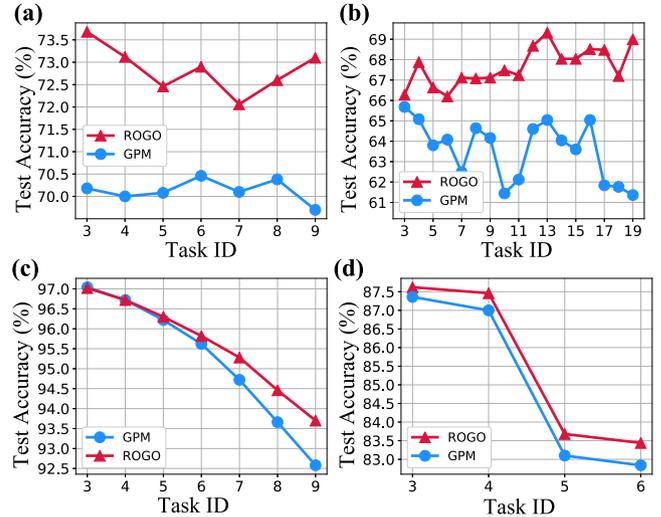
**Table C.15**

Time comparison evaluated on two benchmarks. We use the results reported in Lin et al. (2022) and the time is normalized with respect to GPM.

Datasets	Methods							
	OWM	EWC	HAT	A-GEM	ER_Res	GPM	TRGP	ROGO
CIFAR-100	2.41	1.76	1.62	3.48	1.49	1.00	1.65	1.62
MiniImageNet	-	1.22	0.91	1.79	0.82	1.00	1.34	1.43

### C.5. Time consumption

We report the time consumption of ROGO on two benchmarks compared with relative baselines. TRGP and ROGO are both evaluated on a single NVIDIA GeForce RTX 2080 Ti GPU and we report the results according to Lin et al. (2022). As discussed in Section 5, ROGO takes acceptable extra time compared with GPM on both datasets. For both

**Fig. C.5.** Accuracy evolution of the 2nd task on (a) CIFAR-100 Split, (b) MiniImageNet, (c) PMNIST, and (d) Mixture.**Fig. C.6.** Accuracy evolution of the 4th task on (a) CIFAR-100 Split, (b) MiniImageNet, (c) PMNIST, and (d) Mixture.

datasets, ROGO tasks similar time as TRGP, which is similar to EWC and much less than A-GEM and OWM (see Table C.15).

### C.6. Memory usage

We provide a comparison between TRGP and ROGO on the ratio of the amount of extra parameters concerning the amount of the parameters of the initial network architecture. According to Fig. C.8, TRGP requires at least 200% of the number of extra parameters after learning all tasks on all four benchmarks, while ROGO only stores the representation of the frozen space, which can further be released in the inference phase.

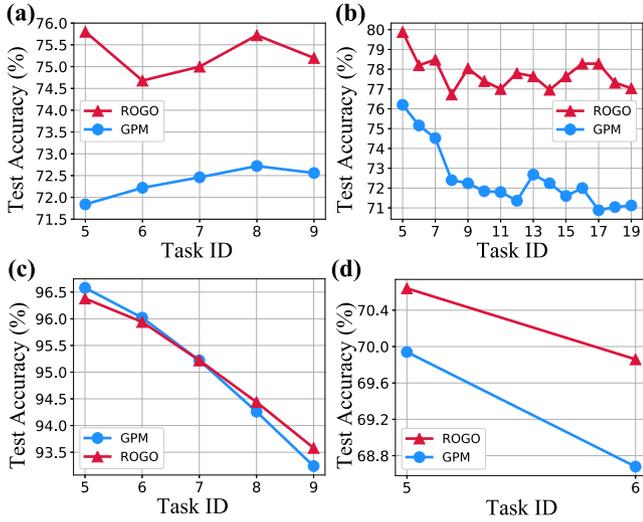


Fig. C.7. Accuracy evolution of the 6th task on (a) CIFAR-100 Split, (b) MiniImageNet, (c) PMNIST, and (d) Mixture.

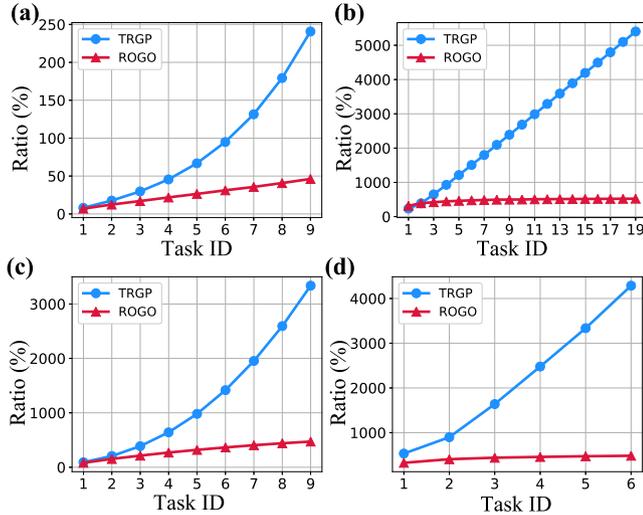


Fig. C.8. Ratio of the amount of extra parameters concerning the amount of the parameters of the initial network architecture on (a) CIFAR100-Split, (b) MiniImageNet, (c) PMNIST, and (d) Mixture.

### C.7. Other results

Here we provide detailed results including the forgetting performance compared with TRGP in Tables C.16 and C.17. Similar to the discussion in Section 4.3, our ROGO framework obtains superior accuracy performance with comparable forgetting under or without the constraint of a fixed network capacity.

Moreover, we illustrate the test accuracy over the task sequence in Fig. C.9. As shown in Fig. C.9-(a) and Fig. C.9-(b), our ROGO-Exp dominates TRGP on both benchmarks relaxing either 80% or T% of the frozen space under an expansion setting. We further compare ROGO with TRGP modified with the same regularization terms. As shown in Fig. C.9-(c) and Fig. C.9-(d), the performance of TRGP drops significantly constrained in a fixed network capacity on both benchmarks.

Following GPM (Saha et al., 2021), we conduct experiments on the CIFAR-100 Sup dataset as well. The implementation details are

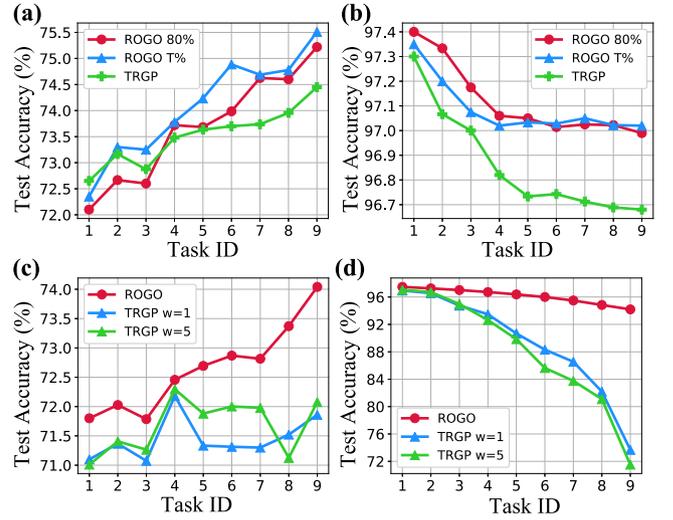


Fig. C.9. Test accuracy after learning each task under an expansion setting on (a) CIFAR-100 Split and (b) PMNIST, and within a fixed network capacity on (c) CIFAR-100 Split and (d) PMNIST.

Table C.16

Comparison of average accuracy and forgetting with TRGP under an expansion setting. The percentages indicate the ratios of the rank of the relaxing space with respect to the frozen space. For each dataset, we mark the best and the second best performance in **bold** and underline.

Method	CIFAR-100 Split		MiniImageNet	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
TRGP	74.46 ± 0.22	-0.42 ± 0.20	61.78 ± 1.94	-1.01 ± 0.58
ROGO-Exp 50%	74.90 ± 0.37	-0.99 ± 0.27	<b>63.46 ± 0.85</b>	<b>0.50 ± 0.39</b>
ROGO-Exp 80%	74.97 ± 0.30	-0.68 ± 0.39	62.57 ± 1.32	0.42 ± 0.57
ROGO-Exp T%	<b>75.34 ± 0.61</b>	<u>-0.63 ± 0.57</u>	<u>62.78 ± 1.00</u>	<u>0.45 ± 0.35</u>
Method	PMNIST		Mixture	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
TRGP	96.34 ± 0.11	-0.58 ± 0.10	83.54 ± 1.15	-0.80 ± 1.10
ROGO-Exp 50%	96.44 ± 0.20	-0.75 ± 0.13	82.45 ± 0.49	<u>-0.74 ± 0.16</u>
ROGO-Exp 80%	96.76 ± 0.15	<u>-0.46 ± 0.08</u>	83.33 ± 0.31	-0.98 ± 0.23
ROGO-Exp T%	<b>97.01 ± 0.08</b>	<b>-0.31 ± 0.05</b>	<b>83.62 ± 0.26</b>	<b>-0.33 ± 0.14</b>

Table C.17

Comparison of average accuracy and forgetting with TRGP within a fixed network capacity. We modify TRGP as TRGP-Reg with similar regularization terms.  $\beta$  indicates the regularization weight. For each dataset, we mark the best and the second best performance in **bold** and underline.

Method	CIFAR-100 Split		MiniImageNet	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
TRGP-Reg $\beta = 1$	72.01 ± 0.30	-1.63 ± 0.20	55.81 ± 2.43	-3.69 ± 2.19
TRGP-Reg $\beta = 5$	71.96 ± 0.40	<u>-1.09 ± 0.37</u>	58.81 ± 2.24	<b>-2.85 ± 1.54</b>
TRGP-Reg $\beta = 50$	<u>72.49 ± 0.09</u>	<b>-0.69 ± 0.24</b>	22.69 ± 0.39	0.01 ± 0.07
ROGO	<b>74.04 ± 0.35</b>	-1.43 ± 0.43	<b>63.66 ± 1.24</b>	<u>-2.44 ± 0.13</u>
Method	PMNIST		Mixture	
	ACC (%)	BWT (%)	ACC (%)	BWT (%)
TRGP-Reg $\beta = 1$	76.57 ± 2.69	-20.69 ± 2.94	73.31 ± 1.05	-11.05 ± 1.43
TRGP-Reg $\beta = 5$	75.50 ± 3.96	-22.41 ± 4.24	74.71 ± 0.85	-9.33 ± 1.05
TRGP-Reg $\beta = 50$	76.56 ± 3.83	-21.63 ± 4.13	17.36 ± 0.86	<b>-0.01 ± 0.03</b>
ROGO	<b>94.20 ± 0.11</b>	<b>-0.09 ± 0.94</b>	<b>77.91 ± 0.45</b>	<u>-4.55 ± 0.36</u>

included in Appendix B.3 as well. Here we report the accuracy performance in Table C.18. Note that the selected baselines for this setting all require extra parameters other than GPM and we directly use the results of the baselines reported in GPM and TRGP. In accordance with the

**Table C.18**

Results of ACC (%) on CIFAR-100 Sup setting. STL is under a non-incremental setting. All baselines require extra network capacity except GPM. We also report the standard deviation result of ROGO for comparison.

Metric	Methods							
	STL*	PNN	DEN	RCL	APD	GPM	TRGP	ROGO
ACC (%)	61.00	50.76	51.10	51.99	56.81	57.72	58.25	<b>58.74 ± 0.60</b>

results on the other four benchmarks discussed in Section 4.2, ROGO outperforms GPM and other related baselines by a significant margin. Particularly, we notice ROGO directly obtains superior performance than TRGP within a fixed network capacity under this setting.

## Appendix D. Algorithm

We present the pseudo-code of our searching strategy here.

---

### Algorithm 2: Relaxing Space Searching

---

**Input:** gradient  $\{g_t^l\}_{l=1}^L$ , frozen space  $\{U_{t-1}^l\}_{l=1}^L$  and thresholds  $\{e_{th}^l, \gamma_t^l\}_{l=1}^L$

**Output:** relaxing space  $\{V_t^l\}_{l=1}^L$

```

1: for  $l \in 1, \dots, L$  do
2:   Construct the significant representation space  $R_{g,t}^l$  from gradients  $g_t^l$  with top- $k$  eigenvectors.
3:    $V_t^l \leftarrow \emptyset$ 
4:   repeat
5:      $d \leftarrow \arg \min_{d \in U_{t-1}^{l,c}} \Theta(d, R_{g,t}^l)$ 
6:     if  $\Theta(d, R_{g,t}^l) \leq \gamma_t^l$  then
7:        $V_t^l \leftarrow V_t^l \cup d$ 
8:        $U_{t-1}^{l,c} \leftarrow U_{t-1}^l \setminus V_t^l$ 
9:     end if
10:  until  $\Theta(d, R_{g,t}^l) > \gamma_t^l$ 
11: end for

```

---

## References

- Bennani, M.A., Doan, T., Sugiyama, M., 2020. Generalisation guarantees for continual learning with orthogonal gradient descent. arXiv preprint arXiv:2006.11942.
- Bulatov, Y., 2011. Notmnist dataset. <http://yaroslavvb.com/upload/notMNIST/>.
- Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M., 2018. Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420.
- Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M., 2019. Continual learning with tiny episodic memories. CoRR, arXiv:1902.10486.
- Chenshen, W., HERRANZ, L., Xialei, L., et al., 2018. Memory replay GANs: Learning to generate images from new categories without forgetting. In: The 32nd International Conference on Neural Information Processing Systems, Montréal, Canada. pp. 5966–5976.
- Choi, Y., El-Khany, M., Lee, J., 2021. Dual-teacher class-incremental learning with data-free generative replay. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3543–3552.
- Cong, Y., Zhao, M., Li, J., Wang, S., Carin, L., 2020. Gan memory with no forgetting. Adv. Neural Inf. Process. Syst. 33, 16481–16494.
- De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T., 2021. A continual learning survey: Defying forgetting in classification tasks. IEEE Trans. Pattern Anal. Mach. Intell. 44 (7), 3366–3385.
- Deng, D., Chen, G., Hao, J., Wang, Q., Heng, P.-A., 2021. Flattening sharpness for dynamic gradient projection memory benefits continual learning. Adv. Neural Inf. Process. Syst. 34, 18710–18721.
- Ebrahimi, S., Meier, F., Calandra, R., Darrell, T., Rohrbach, M., 2020. Adversarial continual learning. In: European Conference on Computer Vision. Springer, pp. 386–402.

- Ehret, B., Henning, C., Cervera, M.R., Meulemans, A., Von Oswald, J., Grewe, B.F., 2020. Continual learning in recurrent neural networks. arXiv preprint arXiv:2006.12109.
- Farajtabar, M., Azizan, N., Mott, A., Li, A., 2020. Orthogonal gradient descent for continual learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, pp. 3762–3773.
- Finn, C., Rajeswaran, A., Kakade, S., Levine, S., 2019. Online meta-learning. In: International Conference on Machine Learning. PMLR, pp. 1920–1930.
- French, R.M., 1997. Pseudo-recurrent connectionist networks: An approach to the ‘sensitivity-stability’ dilemma. Connect. Sci. 9 (4), 353–380.
- Griffiths, T.L., Ghahramani, Z., 2011. The Indian buffet process: An introduction and review. J. Mach. Learn. Res. 12 (4).
- Kao, T.-C., Jensen, K., van de Ven, G., Bernacchia, A., Hennequin, G., 2021. Natural continual learning: success is a journey, not (just) a destination. Adv. Neural Inf. Process. Syst. 34, 28067–28079.
- Kemker, R., McClure, M., Abitino, A., Hayes, T., Kanan, C., 2018. Measuring catastrophic forgetting in neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al., 2017. Overcoming catastrophic forgetting in neural networks. Proc. natl. acad. sci. 114 (13), 3521–3526.
- Kong, Y., Liu, L., Wang, Z., Tao, D., 2022. Balancing stability and plasticity through advanced null space in continual learning. arXiv preprint arXiv:2207.12061.
- Krizhevsky, A., Hinton, G., 2009. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer.
- Kumar, A., Chatterjee, S., Rai, P., 2021. Bayesian structural adaptation for continual learning. In: International Conference on Machine Learning. PMLR, pp. 5850–5860.
- Kurle, R., Cseke, B., Klushyn, A., Van Der Smagt, P., Günnemann, S., 2019. Continual learning with bayesian neural networks for non-stationary data. In: International Conference on Learning Representations.
- LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.
- Lin, S., Yang, L., Fan, D., Zhang, J., 2022. TRGP: Trust region gradient projection for continual learning. arXiv preprint arXiv:2202.02931.
- Liu, H., Liu, H., 2022. Continual learning with recursive gradient optimization. arXiv preprint arXiv:2201.12522.
- Lopez-Paz, D., Ranzato, M., 2017. Gradient episodic memory for continual learning. Adv. neural inf. process. syst. 30.
- Mallya, A., Lazebnik, S., 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7765–7773.
- McCloskey, M., Cohen, N.J., 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of Learning and Motivation, Vol. 24. Elsevier, pp. 109–165.
- Mirzadeh, S.I., Farajtabar, M., Pascanu, R., Ghasemzadeh, H., 2020. Understanding the role of training regimes in continual learning. Adv. Neural Inf. Process. Syst. 33, 7308–7320.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., 2011. Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning.
- Ng, H.-W., Winkler, S., 2014. A data-driven approach to cleaning large face datasets. In: 2014 IEEE International Conference on Image Processing. ICIP, IEEE, pp. 343–347.
- Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Netw. 113, 54–71.
- Ratcliff, R., 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. Psychol. rev. 97 (2), 285.
- Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R., 2016. Progressive neural networks. arXiv preprint arXiv:1606.04671.
- Saha, G., Garg, I., Roy, K., 2021. Gradient projection memory for continual learning. arXiv preprint arXiv:2103.09762.
- Serra, J., Suris, D., Miron, M., Karatzoglou, A., 2018. Overcoming catastrophic forgetting with hard attention to the task. In: International Conference on Machine Learning. PMLR, pp. 4548–4557.
- Shin, H., Lee, J.K., Kim, J., Kim, J., 2017. Continual learning with deep generative replay. Adv. neural inf. process. syst. 30.
- Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C., 2011. The german traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 International Joint Conference on Neural Networks. IEEE, pp. 1453–1460.
- Thrun, S., Mitchell, T.M., 1995. Lifelong robot learning. Robot. auton. syst. 15 (1–2), 25–46.

- Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al., 2016. Matching networks for one shot learning. *Adv. neural inf. process. syst.* 29.
- Wang, S., Li, X., Sun, J., Xu, Z., 2021. Training networks in null space of feature covariance for continual learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 184–193.
- Xiao, H., Rasul, K., Vollgraf, R., 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.
- Yoon, J., Kim, S., Yang, E., Hwang, S.J., 2019. Scalable and order-robust continual learning with additive parameter decomposition. *arXiv preprint arXiv:1902.09432*.
- Yoon, J., Yang, E., Lee, J., Hwang, S.J., 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.
- Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S., Weijer, J.v.d., 2020. Semantic drift compensation for class-incremental learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 6982–6991.
- Zeng, G., Chen, Y., Cui, B., Yu, S., 2019. Continual learning of context-dependent processing in neural networks. *Nat. Mach. Intell.* 1 (8), 364–372.
- Zhu, F., Zhang, X.-Y., Wang, C., Yin, F., Liu, C.-L., 2021. Prototype augmentation and self-supervision for incremental learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 5871–5880.