

# Supplementary Materials: Introducing Common Null Space of Gradients for Gradient Projection based Continual Learning

## A PROOF OF THE PROPOSITION

### A.1 Proposition 1

**Proposition 1.** Suppose we perform SVD on a matrix  $G_t \in \mathbb{R}^{tm \times n}$ :

$$\text{SVD}(G_t) = [\tilde{U}_t, \bar{U}_t] \begin{bmatrix} \Sigma_t & 0 \\ 0 & 0 \end{bmatrix} [\tilde{V}_t, \bar{V}_t]^\top. \quad (1)$$

Where  $U_t = [\tilde{U}_t, \bar{U}_t] \in \mathbb{R}^{tm \times tm}$  and  $V_t = [\tilde{V}_t, \bar{V}_t] \in \mathbb{R}^{n \times n}$  are the left singular matrix and right singular matrix respectively. Then we have the proposition that  $I - \tilde{V}_t \tilde{V}_t^\top$  is the null space of  $G_t$ .

*Proof.* We prove the proposition by verifying whether the product of  $G_t$  and  $I - \tilde{V}_t \tilde{V}_t^\top$  equals zero matrix. We know that  $\tilde{V}_t \in \mathbb{R}^{n \times \min\{tm, n\}}$  and  $\tilde{V}_t \tilde{V}_t^\top \in \mathbb{R}^{n \times n}$ , then we have:

$$\begin{aligned} G_t(I - \tilde{V}_t \tilde{V}_t^\top) &= (\tilde{U}_t \Sigma_t \tilde{V}_t^\top)(I - \tilde{V}_t \tilde{V}_t^\top) \\ &= (\tilde{U}_t \Sigma_t \tilde{V}_t^\top) - (\tilde{U}_t \Sigma_t \tilde{V}_t^\top) \tilde{V}_t \tilde{V}_t^\top \\ &= \tilde{U}_t \Sigma_t \tilde{V}_t^\top - \tilde{U}_t \Sigma_t \tilde{V}_t^\top = 0. \end{aligned} \quad (2)$$

Thus,  $I - \tilde{V}_t \tilde{V}_t^\top$  is the null space of  $G_t$  according to Definition 3.

### A.2 Proposition 2

**Proposition 2.** Suppose we perform SVD on a matrix  $H_t \in \mathbb{R}^{n \times (h_{t-1} + k_t)}$ :

$$\text{SVD}(H_t) = U_{t,H_t} \begin{bmatrix} \Sigma_{t,H_t} & 0 \\ 0 & 0 \end{bmatrix} V_{t,H_t}^\top. \quad (3)$$

Where  $U_{t,H_t} = [\tilde{U}_{t,H_t}, \bar{U}_{t,H_t}] \in \mathbb{R}^{n \times n}$  and  $V_{t,H_t} = [\tilde{V}_{t,H_t}, \bar{V}_{t,H_t}] \in \mathbb{R}^{(h_{t-1} + k_t) \times (h_{t-1} + k_t)}$ . Then we have the proposition that  $\tilde{U}_{t,H_t}$  is the range space of  $H_t$ .

*Proof.* We prove Proposition 2 by examining whether  $H_t$  can be represented by  $\tilde{U}_{t,H_t}$ . We denote the product of  $\Sigma_{t,H_t}$  and  $\tilde{V}_{t,H_t}^\top$  as  $Q_{t,H_t}$ , namely  $Q_{t,H_t} = \Sigma_{t,H_t} \tilde{V}_{t,H_t}^\top \in \mathbb{R}^{\min\{tm, n\} \times n}$ . Then, we have

$$H_t = \tilde{U}_{t,H_t} \Sigma_{t,H_t} \tilde{V}_{t,H_t}^\top = \tilde{U}_{t,H_t} Q_{t,H_t}. \quad (4)$$

Thus,  $\tilde{U}_{t,H_t}$  is the range space of  $H_t$  according to Definition 2.

## B BASIC CONCEPTS IN VECTOR SPACE

**Definition 1** (Subspace) Suppose vectors  $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^m$ , all the linear combinations of these vectors constitute a subspace, which is called the span of  $\{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ :

$$\text{span}\{\mathbf{a}_1, \dots, \mathbf{a}_n\} = \left\{ \sum_{j=1}^n \beta_j \mathbf{a}_j : \beta_j \in \mathbb{R} \right\}. \quad (5)$$

**Definition 2** (Range Space (Golub and Van Loan [1], pg.64)) Suppose  $A \in \mathbb{R}^{m \times n}$  is a matrix. The range space of  $A$  is defined by:

$$\mathcal{R}(A) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = A\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^n\}. \quad (6)$$

Where  $\mathcal{R}(A)$  represents the range space of  $A$ .

**Definition 3** (Null Space (Golub and Van Loan [1], pg.64)) Suppose  $A \in \mathbb{R}^{m \times n}$  is a matrix. The null space of  $A$  is defined by:

$$\mathcal{N}(A) = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = 0\}. \quad (7)$$

Where  $\mathcal{N}(A)$  represents the null space of  $A$ .

**Lemma 1** (Relation between  $\mathcal{R}(A)$  and  $\mathcal{N}(A)$  (Meyer [2], pg.405)) Suppose  $\mathcal{R}(A)$  and  $\mathcal{N}(A)$  are the range space and null space of matrix  $A \in \mathbb{R}^{m \times n}$  respectively, then we have:

$$\mathcal{R}(A)^\perp = \mathcal{N}(A^\top); \quad \mathcal{N}(A)^\perp = \mathcal{R}(A^\top). \quad (8)$$

## C ALGORITHM OF THE COLLABORATIVE FRAMEWORK FOR FSCLP AND GPCNS

---

**Algorithm 1** Algorithm for FSCLP + GPCNS

---

**Require:** Datasets  $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_T\}$  for each tasks; A neural network  $f_W$  parameterized by  $W$  with  $L$  layers, learning rate  $\eta$ , threshold  $\epsilon$  and scale coefficient  $\alpha$ .

**Ensure:**  $f_W, S_T, \Lambda_T$ ;

initialization  $f_W$

**for**  $t = 1, 2, \dots, T$  **do**

**while** not converged **do**

$B_t \sim \mathcal{D}_t$

$\nabla \mathcal{L}_t \leftarrow \text{Optimizer}(B_t, f_W)$

$\nabla \tilde{\mathcal{L}}_t \leftarrow \text{Project}(\nabla \mathcal{L}_t, S_{t,C_t}, \Lambda_{t,C_t})$

$W_t \leftarrow W_t - \eta \nabla \tilde{\mathcal{L}}_t$

**end while**

**for**  $l = 1, 2, \dots, L$  **do**

$G_t^l \leftarrow \text{VerticalStack}(G_{t-1}^l, \nabla \tilde{\mathcal{L}}_t^l)$

$\Sigma_t^l, (\tilde{V}_t^l)^\top \leftarrow \text{SVD}(G_t^l)$

$\tilde{V}_t^l \leftarrow \text{Transpose}((\tilde{V}_t^l)^\top)$

$k_t^l \leftarrow \text{Criteria}(\Sigma_t^l, \epsilon_t^l)$

$\tilde{V}_t^l \leftarrow \tilde{V}_t^l[:, 0 : k_t^l]$

        // Construct Feature Space

$\bar{B}_t \sim \mathcal{D}_t$

$R_t^l \leftarrow \text{Forward}(\bar{B}_t, f_W)$

$M_t^l \leftarrow \text{FSCLP}(R_t^l)$

        // Space Merging

$C_t^l \leftarrow \text{HorizontalStack}(\tilde{V}_t^l, M_t^l)$

$\bar{U}_{t,C_t}^l, \Sigma_{t,C_t}^l \leftarrow \text{SVD}(C_t^l)$

$r_t^l \leftarrow \text{Criteria}(\Sigma_{t,C_t}^l, \epsilon_t^l)$

$S_{t,C_t}^l \leftarrow \bar{U}_{t,C_t}^l[:, 0 : r_t^l]$

$\Lambda_{t,C_t}^l \leftarrow \text{Scaling}(\Sigma_{t,C_t}^l, \alpha)$

**end for**

**end for**

---

## D SIZE OF REPRESENTATION AND GRADIENTS

### D.1 Matrix Transformed from Tensor

We reshape the gradient matrix on each convolutional layer from four dimensions to two dimensions, namely  $\nabla \mathcal{L}_t \in \mathbb{R}^{d_O \times d_I \times d_K \times d_K} \Rightarrow \nabla \mathcal{L}_t \in \mathbb{R}^{d_O \times (d_I \times d_K \times d_K)}$ . Where  $d_O$  and  $d_I$  are output and input channels, respectively.  $d_K \times d_K$  is the kernel size. The gradient matrix mentioned in the main paper are all two-dimensional, and we denote their size as  $\nabla \mathcal{L}_t \in \mathbb{R}^{m \times n}$  for the convenience of representation.

### D.2 Size of Representation and Gradient on Each Layer

In Table 1, we demonstrate the sizes of representation matrices and gradient matrices constructed to determine the projection space when applying a ResNet-18 as the backbone for continual learning on 20-Split MiniImageNet. The representation matrices carry feature information, and the method to compute representation is the same as GPM, CGP, TRGP, and SGP. We can find that the size of the gradient matrix is much smaller than that of representation matrix on each layer. Therefore, constructing orthogonal spaces through gradient information requires less training time and memory overhead.

**Table 1: The sizes of the representation matrices and gradient matrices for each layer when applying a ResNet-18 on MiniImageNet. Where Repres and Grad are the abbreviation of Representation and Gradient respectively. Pixels in Repres is the number of elements in the representation matrix, and Pixels in Grad is the number of elements in the gradient matrix. R/G represents the pixel ratio of representation and gradient.**

Layer	Size of Repres	Size of Grad	Pixels in Repres	Pixels in Grad	Repres/Grad
1	27×17640	20×27	476280	540	882.0
2	180×17640	20×180	3175200	3600	882.0
3	180×17640	20×180	3175200	3600	882.0
4	180×17640	20×180	3175200	3600	882.0
5	180×17640	20×180	3175200	3600	882.0
6	180×4410	40×180	793800	7200	110.25
7	360×4410	40×360	1587600	14400	110.25
8	20×4410	40×20	88200	800	110.25
9	360×4410	40×360	1587600	14400	110.25
10	360×22050	40×360	7938000	14400	551.25
11	360×6050	80×360	2178000	28800	75.625
12	720×6050	80×720	4356000	57600	75.625
13	40×6050	80×40	242000	3200	75.625
14	720×12100	80×720	8712000	57600	151.25
15	720×12100	80×720	8712000	57600	151.25
16	720×3600	160×720	2592000	115200	22.5
17	1440×3600	160×1440	5184000	230400	22.5
18	80×3600	160×80	288000	12800	22.5
19	1440×3600	160×1440	5184000	230400	22.5
20	1440×3600	160×1440	5184000	230400	22.5

## E COMPUTATIONAL ISSUE AND SOLUTION

### E.1 Computational Issue

In the main paper, we propose GPCNS based on a novel design idea, which constructs projection spaces with the help of gradient information from previous tasks. We construct a common gradient matrix  $G_t$  by stacking the projected gradients under all previous tasks vertically, and the first dimension of  $G_t \in \mathbb{R}^{tm \times n}$  will increase with increasing tasks. Since the size of gradient matrix is much smaller than representation matrix, GPCNS has a shorter training time and occupies less memory in the experiments. However, GPCNS needs to store the gradient matrix for each task, which will cause its advantage in computational cost to diminish or even disappear as the number of tasks increases.

In most continual learning settings, the number of tasks is less than or equal to 20. In this scenario, GPCNS is lightweight. Taking the experiment on MiniImageNet as an example, representation matrices on each layer are more than 20 times larger than the gradient matrices, with the largest gap even reaching 882 times (see Table 1). Therefore, even if the gradients under all tasks are stored at last task, the size of these gradients concatenated together is still smaller than the representation matrix of one task.

Nevertheless, we still need to consider situations when the number of tasks is extraordinarily large. Thus, we propose the following solution.

## E.2 Solution and Future Work

A possible solution is to construct a novel type of common gradient matrix that

$$G_t = \nabla \mathcal{L}_t^* \bar{V}_1 \bar{V}_2 \cdots \bar{V}_{t-1} \quad (9)$$

instead of  $G_t$  in the main paper. Under such a condition, we can obtain that  $\bar{V}_t$  is the null space of  $G_t$  if SVD is performed on  $G_t$  and  $V_t = [\bar{V}_t, \bar{V}_t^\perp]$  is obtained according to Eq.(1). Furthermore,  $(\bar{V}_1 \bar{V}_2 \cdots \bar{V}_{t-1}) \bar{V}_t$  is the common null space of  $\nabla \mathcal{L}_\tau^*$ ,  $\tau = 1, 2, \dots, t$ , and the reason is as follow. From Eq.(9), we have

$$\begin{cases} G_1 \bar{V}_1 = \nabla \mathcal{L}_1^* \bar{V}_1 = \mathbf{0} \\ G_2 \bar{V}_2 = (\nabla \mathcal{L}_2^* \bar{V}_1) \bar{V}_2 = \mathbf{0} \\ \vdots \\ G_t \bar{V}_t = (\nabla \mathcal{L}_t^* \bar{V}_1 \bar{V}_2 \cdots \bar{V}_{t-1}) \bar{V}_t = \mathbf{0} \end{cases} \quad (10)$$

By multiplying the same term on both sides of the sub-equation in each row, we can obtain

$$\begin{cases} \nabla \mathcal{L}_1^* \bar{V}_1 (\bar{V}_2 \cdots \bar{V}_t) = \mathbf{0} \cdot (\bar{V}_2 \cdots \bar{V}_t) \\ \nabla \mathcal{L}_2^* (\bar{V}_1 \bar{V}_2) (\bar{V}_3 \cdots \bar{V}_t) = \mathbf{0} \cdot (\bar{V}_3 \cdots \bar{V}_t) \\ \vdots \\ \nabla \mathcal{L}_{t-1}^* (\bar{V}_1 \cdots \bar{V}_{t-1}) \bar{V}_t = \mathbf{0} \cdot \bar{V}_t \\ \nabla \mathcal{L}_t^* (\bar{V}_1 \cdots \bar{V}_{t-1} \bar{V}_t) = \mathbf{0} \end{cases} \quad (11)$$

Therefore,  $(\bar{V}_1 \bar{V}_2 \cdots \bar{V}_{t-1}) \bar{V}_t$  is the common null space of  $\nabla \mathcal{L}_\tau^*$ ,  $\tau = 1, 2, \dots, t$  and  $\bar{V}_t$  is the null space of  $G_t$ .

The memory overhead will be significantly reduced since there is no stacking operation of gradient matrices in Eq.(9). Nevertheless, the above solution will result in at least three major problems. (1) The size of  $\bar{V}_t$  will be altered, which also leads to variations on the size of the matrices that depends on  $\bar{V}_t$  (e.g.  $H_t$  and  $C_t$ ). (2) The calculation error will be transferred and amplified in multiple matrix multiplication operations, which results in a decrease on ACC. (3) The second dimension of matrix  $\bar{V}_t$  will continue to decrease until it reaches zero. The above issues make the situation quite complex, which will be considered as a part of the next study.

## F ADDITIONAL RESULTS AND ANALYSIS

In Figure 1, we demonstrate the ablation experiment on gradient scaling in task level. The results indicate that gradient scaling has a certain improvement on test accuracy in most tasks, while the increase is limited to a small margin.

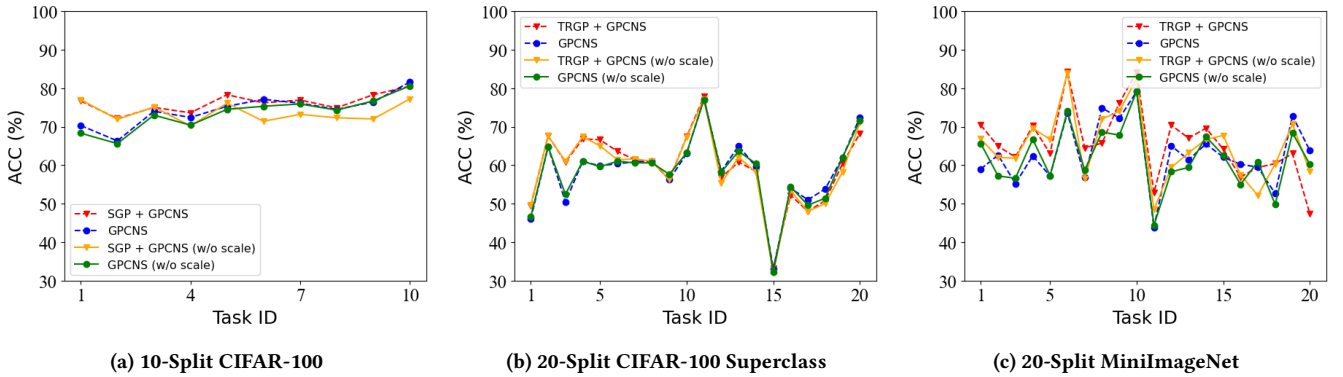


Figure 1: Task level ablation study on gradient scaling.

## G DATASET STATISTICS AND MORE EXPERIMENTAL DETAILS

We give the statistics of three datasets applied to conduct experiments in Table 2. In addition, the settings of hyperparameters for all the considered methods are demonstrated in Table 3. Where CIFAR-100, Superclass and MiniImageNet denote 10-Split CIFAR-100, 20-Split CIFAR-100 Superclass and 20-Split MiniImageNet respectively.

**Table 2: Statistics of three datasets used in the experiment.**

	10-Split CIFAR-100	20-Split CIFAR-100 Superclass	20-Split MiniImageNet
Total Number of Tasks	10	20	20
Total Number of Classes	100	100	100
Size of Input Data	3×32×32	3×32×32	3×84×84
Number of Classes / Task	10	5	5
Sample Size of Training Set / Task	4750	2375	2450
Sample Size of Valid Set / Task	250	125	50
Sample Size of Test Set / Task	1000	500	500

**Table 3: List of hyperparameter settings in baseline approaches and our methods. Where 'lr' represents the initial learning rate, and  $n_s$  is the sample size sampled from the previous tasks in order to construct the projection space for current task.**

Methods	Hyperparameter Settings
Multitask	lr: 0.05 (CIFAR-100), 0.01 (Superclass), 0.1 (MiniImageNet)
OWM	lr: 0.01 (CIFAR-100), 0.1 (MiniImageNet)
A-GEM	lr : 0.05 (CIFAR-100, Superclass), 0.1 (MiniImageNet) memory size (samples) : 2000 (CIFAR-100, Superclass), 500 (MiniImageNet)
ER_Res	lr: 0.05 (CIFAR-100, Superclass), 0.1 (MiniImageNet)
Adam-NSCL	lr: $10^{-4}$ (CIFAR-100, Superclass), $5 \times 10^{-5}$ (MiniImageNet)
GPM	lr: 0.01 (CIFAR-100, Superclass), 0.1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)
FS-DGPM	lr, $\eta_3$ : 0.01 (CIFAR-100, Superclass), 0.1 (MiniImageNet) lr for sharpness, $\eta_1$ : 0.001 (CIFAR-100), 0.01 (Superclass, MiniImageNet) lr for DGPM, : 0.01 (CIFAR-100, Superclass, MiniImageNet) memory size (samples) : 1000 (CIFAR-100, Superclass, MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)
CGP	lr: 0.04 (CIFAR-100), 0.03 (Superclass), 0.1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)
TRGP	lr: 0.01 (CIFAR-100, Superclass), 0.1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)
SGP	lr: 0.05 (CIFAR-100), 0.01 (Superclass), 0.1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet) $\alpha$ : 5 (CIFAR-100), 3 (Superclass), 1 (MiniImageNet)
GPCNS	lr: 0.05 (CIFAR-100), 0.01 (Superclass), 0.1 (MiniImageNet) $\alpha$ : 5 (CIFAR-100), 4.5 (Superclass), 3 (MiniImageNet)
GPM + GPCNS	lr: 0.05 (CIFAR-100), 0.01 (Superclass), 0.1 (MiniImageNet) $\alpha$ : 1.5 (CIFAR-100), 4.5 (Superclass), 1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)
TRGP + GPCNS	lr: 0.05 (CIFAR-100), 0.01 (Superclass), 0.1 (MiniImageNet) $\alpha$ : 1.5 (CIFAR-100), 4.5 (Superclass), 1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)
SGP + GPCNS	lr: 0.05 (CIFAR-100), 0.01 (Superclass), 0.1 (MiniImageNet) $\alpha$ : 1.5 (CIFAR-100), 4.5 (Superclass), 1 (MiniImageNet) $n_s$ : 125 (CIFAR-100, Superclass), 100 (MiniImageNet)

## H SOFTWARE AND HARDWARE

We implemented GPCNS and FSCLP + GPCNS in python (version 3.10.9) with pytorch (version 1.13.1) and torchvision (version 0.14.1) libraries. All the experiments were performed on a single machine with Ubuntu 18.04, 40 Intel(R) Xeon(R) Silver 4210R CPUs @ 2.40GHz and a NVIDIA A100 GPU (CUDA Version: 11.6).

## REFERENCES

- [1] Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press. 64 pages.
- [2] Carl D Meyer. 2000. *Matrix analysis and applied linear algebra*. Vol. 71. SIAM.