A APPENDIX

B PROOF OF INSIGHT WITH BALANCING STABILITY AND PLASTICITY

Lemma 1 *Singular value decomposition (SVD): For matrix $A_{m,n}$, we can factorize it into three matrices and obtain matrix $U_{m,l}$, $\Sigma_{l,l}$, and $V_{l,n}$, where $U_{m,l}$ and $V_{l,n}$ are orthogonal matrices, $\Sigma_{l,l}$ contains the sorted singular value along its main diagonal:*

$$A_{m,n} = U_{m,l}\Sigma_{l,l}{V_{l,n}}^T. \tag{22}$$

Theorem 1 *For any embedding sequences x_t, embedded from samples of task t, using singular value decomposition (SVD) in lemma 1, we can obtain matrix U_t, Σ_t, and V_t. Then, we randomly split V_t along the column dimension into two parts: $[V_t^1, V_t^2]$. Because V_t is an orthogonal matrix, we can have:*

$$V_t^l {V_t^l}^T = [V_t^1 V_t^2] \begin{bmatrix} V_t^{1T} \\ V_t^{2T} \end{bmatrix} = V_t^1 V_t^{1T} + V_t^2 V_t^{2T} = I. \tag{23}$$

On Plasticity: For any matrix V and gradient g_{t+1} when learning task $t+1$, we have the gradient after projection on V as $g_{t+1}^{'}$, and calculate the cosine similarity between g_{t+1} and $g_{t+1}^{'}$:

$$\begin{aligned} < g_{t+1}^{'}, g_{t+1} > &=< g_{t+1}VV^T, g_{t+1} > \\ &= \text{vec}(g_{t+1}VV^TI)^T\text{vec}(g_{t+1}) \\ &= \text{vec}(g_{t+1}V)^T(I)(I \times V)\text{vec}(g_{t+1}) \\ &= \text{vec}(g_{t+1}V)^T\text{vec}(g_{t+1}V) \\ &\geq 0, \end{aligned} \tag{24}$$

which means that whatever V is, as if g_{t+1} and V both are not equal to zero, after projection on V, the new gradient is always positive. Thus, the model is always learning new knowledge. The only distinction is the gradient direction, which can be measured by calculation of cosine similarity between gradient before projection and after projection. We set gradient project matrix as V_t^2 and have:

$$\begin{aligned} < g_{t+1}^{'}, g_{t+1} > &=< g_{t+1}V_t^2V_t^{2T}, g_{t+1} > \\ &=< g_{t+1}(I - V_t^1V_t^{1T}), g_{t+1} > \\ &=< g_{t+1} - g_{t+1}V_t^1V_t^{1T}, g_{t+1} > \\ &=< g_{t+1}, g_{t+1} > - < g_{t+1}V_t^1V_t^{1T}, g_{t+1} > \\ &\leq < g_{t+1}, g_{t+1} > . \end{aligned} \tag{25}$$

From the inequality, if projection matrix V_t^2 is not an identity matrix, which means that matrix V_t^1 is not a zero matrix, the direction of gradient after projection always has an angle with the direction of the original gradient, incurring that decrease of loss function becomes slow.

For further research on the relationship between matrix V_t^1 and decreased speed of loss function, we mainly focus on two situations for value of V_t^1:

(1).$V_t^1 = 0$, we have

$$< g_{t+1}^{'}, g_{t+1} >=< g_{t+1}, g_{t+1} > . \tag{26}$$

In this situation, it equals that we do not operate on the gradient, and parameters are updated normally. When $V_t^1 = 0$, it has $V_t^2 = V_t$. In fact, the same phenomenon has been shown in the previous situation that $V_t^2 = V_t$.

(2).$V_t^1 = V_t$, we have:

$$\begin{aligned} < g_{t+1}^{'}, g_{t+1} > &=< g_{t+1}, g_{t+1} > - < g_{t+1}V_tV_t^T, g_{t+1} > \\ &=< g_{t+1}, g_{t+1} > - < g_{t+1}, g_{t+1} > \\ &= O. \end{aligned} \tag{27}$$

In this situation, it equals that we freeze the network update process, and trainable parameters are stable and not changed. Thus, the network will not adopt any other new knowledge. When $V_t^1 = V_t$, it has $V_t^2 = 0$. In fact, the same phenomenon has been shown in the previous situation that $V_t^2 = 0$.

In conclusion, with V_t^1 changing from 0 to V_t, the decreased speed of the loss function becomes more and more slow, leading to worse plasticity. However, under this trend, V_t^2 is changing from V_t to 0, giving the anti-forgetting more and more strength. We can recognize that the essence of the gradient projection method is a kind of trade-off strategy between plasticity and stability. However, different from other dilemmas, it has an optimal solution, which is projecting gradient in the direction orthogonal to the subspace spanned by the old inputs, which can not only own the best ability of anti-forgetting, but also have minimal damage to plasticity.

C METHOD OF UPDATING PROJECTION MATRIX

We update our projection matrix $V_{t,0}$ like GPM (Saha et al., 2021), which is detailed described as follows. Assume that we have sampled embedding sequences from current task samples x_t and trained prompts p_t. Here, t is the task identifier. We utilize Principal Component Analysis (PCA) to compress and align the dimensions of x_t and p_t. Then, we element-wise add x_t and p_t to obtain s_t. Besides that, we set a threshold ϵ.

For task #1 training, we perform SVD on s_1 as $s_1 = U_1 \Sigma_1 {V_1}^T$. We collect the minimum former l columns of V_1 as matrix $L = [v_{11}, v_{12}, ..., v_{1l}]$ according to the following criteria:

$$||s_{1l}||_F^2 \geq \epsilon ||s_1||_F^2. \tag{28}$$

Here, $||.||_F$ is the Frobenius norm of the matrix and ϵ $(0 < \epsilon \leq 1)$ is the threshold hyperparameter. $V_{1,0}$ can be obtained by $V_{1,0} V_{1,0}^T = I - LL^T$.

For task #2 training, before performing SVD and subsequent former-rank approximation, we eliminate the common directions in s_2 which are already present in L, so that newly added column vectors are unique and orthogonal to the existing column vectors. Thus, we perform the step $\hat{s}_2 = s_2 - LL^T s_2$. Afterward, SVD is performed on $\hat{s}_2 (= \hat{U}_2 \hat{\Sigma}_2 \hat{V}_2^T)$ and former m new columns of $\hat{V}_2$ are chosen with minimum value of m satisfying the following criteria:

$$||LL^T s_2||_F^2 + ||\hat{s}_{2m}||_F^2 \geq \epsilon ||s_2||_F^2. \tag{29}$$

Here, L is updated by adding new column vectors as $[v_{11}, v_{12}, ..., v_{1l}, \hat{v}_{21}, \hat{v}_{22}, ..., \hat{v}_{2m}]$. Then, we can update $V_{1,0}$ to $V_{2,0}$ according to $V_{2,0} V_{2,0}^T = I - LL^T$. Once the update is complete we move on to the next task and repeat the same procedure as in task #2.

D COMPUTATION COST FOR MAINTAINING THE ORTHOGONALITY OF THE TASK SUBSPACES

In this section, we will discuss the added computation cost for maintaining the orthogonality of the task subspaces under the following situations.

D.1 LARGER MODELS

If we change the backbone from a smaller one to a larger one, it could have different results of added computation cost for distinct tuning paradigms. i) For prompt-tuning, because we only prepend the prompt into the first transformer layer, the added computation could be omitted. ii) For prefix-tuning, larger models usually mean more network layers or wider input dimensions, and we need to expand the prefix-inserted layer or prefix width, which is the origination of the added computation cost. For expanding the prefix-inserted layer, each layer can have a nearly similar computation cost if the number of samples is the same. Thus, we can conclude that the added computation cost can be modeled as an approximate linear function with the layer numbers of the backbone. Similarly, the same conclusion can also be drawn from expanding the prefix width.

D.2 INCREASED NUMBER OF TASKS

Observing the training processes of multi-datasets, we can empirically summarize that in each task, the number of newly added column vectors of the projection matrix is constant in a certain range. As the added computation cost is mainly focused on i) calculation of the projection matrix and ii) multiplication between the projection matrix and its transpose, we can see that although it could not appear exponential explosion, it is still a potential risk in our method with the increased number of tasks.

E GRADIENT PROJECTION BASED ON PREFIX-TUNING PARADIGM

In this section, we prove that the gradient projection method can be utilized in prefix-tuning with mathematical deduction.

Distinct from prompt-tuning paradigm, prefix-tuning only prepends prefixes in key vector and value vector, without query vector of prepended transformer layer. Additionally, different from prompt usually only prepended in the first transformer layer, prefix can be prepended in any transformer layers. These advantages help models based on prefix-tuning own a better performance than those based on prompt-tuning both in natural language processing and computer vision.

For baseline based on prefix-tuning, if we want to preserve old knowledge, we need to realize:

$$f_\theta(p_{t,l}, x_{t,l}) = f_\theta(p_{t+1,l}, x_{t,l}). \tag{30}$$

f_θ refers to ViT model, $x_{t,l}$ denotes inputs at task t in layer l, $p_{t,l}$ and $p_{t+1,l}$ represents the prefixes trained at task t and prepended in layer l and the prefixes trained at task $t+1$ and prepended in layer l respectively.

Assuming that a set of prefixes have been trained at task $t+1$, and we input samples from task t. Now, we prepend prefix in key vector, and have:

$$Q_{t,l} = W_{q,l} x_{t,l}, \tag{31}$$

$$K_{t,l} = \begin{bmatrix} p_{t+1,l} \\ W_{k,l} x_{t,l} \end{bmatrix}, \tag{32}$$

where, $W_{q,l}$ and $W_{k,l}$ are weights of ViT, frozen and unchanged. With Eq.(3), we have the results that t-th task samples on $t+1$-th model. We mainly focus on the part:

$$Q_{t,l} K_{t,l}^T = W_{q,l} x_{t,l} \begin{bmatrix} p_{t+1,l}^T & (W_{k,l} x_{t,l})^T \end{bmatrix} = \begin{bmatrix} W_{q,l} x_{t,l} p_{t+1,l}^T & W_{q,l} x_{t,l} x_{t,l}^T W_{k,l}^T \end{bmatrix}. \tag{33}$$

As stable item $W_{q,l} x_{t,l} x_{t,l}^T W_{k,l}^T$, we only focus on the item $W_{q,l} x_{t,l} p_{t+1,l}^T$. Changing $p_{t+1,l}^T$ with $p_{t,l}^T$, we can obtain the results that t-th task samples on t-th model. Because our aim is making $W_{q,l} x_{t,l} p_{t+1,l}^T$ equal to $W_{q,l} x_{t,l} p_{t,l}^T$, considering that $W_{q,l}$ is frozen, our final aim can be simplified as:

$$x_{t,l} p_{t+1,l}^T = x_{t,l} p_{t,l}^T, \tag{34}$$

which has the same form as Eq.(8), meaning that we can also achieve Eq.(34) by the gradient projection method. Thus, we can draw the conclusion that the gradient projection method could also help models based on prefix-tuning to resist forgetting.

F METRICS

Two metrics: Average Accuracy (simplified as accuracy/ACC) and Forgetting (simplified as FOR) are used to evaluate the performance. We use average accuracy metric, for averaging the classification accuracy of all classes. We adopt forgetting metric to indicate the average loss of accuracy of past tasks after learning a new task. Formally, average accuracy and forgetting are defined as:

$$\text{Average Accuracy} = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}, \tag{35}$$

$$\text{Forgetting} = \frac{1}{T-1}\sum_{i=1}^{T-1} A_{T,i} – \max(A_{j,i})_{j\in[i,T-1]}, \tag{36}$$

where T is the number of tasks, $A_{T,i}$ is the accuracy of i-th task samples on the T-th model, and $A_{j,i}$ is the accuracy of i-th task samples on the j-th model.

G EXPERIMENTAL DETAILS

Consistent with previous works (Wang et al., 2022c;b; Smith et al., 2023), we use ViT B/16 (Dosovitskiy et al., 2020) pre-trained on ImageNet-21K as our image encoder, which is kept frozen during training. We train and test on one A6000-48GB GPU for baselines and our method. We set the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

For hyperparameters, in L2P-PGP, we set $\epsilon = 0.50$ for extraction of prompt gradient projection matrix and $\epsilon = 0.97$ for key gradient projection matrix. While in DualPrompt-PGP, we set $\epsilon = 0.50$ for extraction of prompt gradient projection matrix. To accelerate the speed of gradient projection matrix extraction and reduce the training space, we add PCA into our process, which can be used to compress the sampled feature space.

In comparison with L2P and L2P-PGP, for 10/20-Split-CIFAR100, and 10-Split-TinyImageNet, we both train the network for 5 epochs with batch size of 16 and prompt length is set at 5, while we both set epochs as 50, batch size as 16, and prompt length as 30 for 10-Split-ImageNet-R.

In comparison with DualPrompt and DualPrompt-PGP, for 10/20-Split-CIFAR100, we train the network for 20 epochs with batch size of 24, and expert prompt length is set at 5. While we both set epochs as 5, batch size as 24, and expert prompt length as 5 for 10-Split-TinyImageNet, epochs as 50 and batch size as 24 for 10-Split-ImageNet-R with expert prompt length at 20. Besides that, in all benchmark datasets, the general prompt length is set at 5 and the prompt-inserted locations are kept the same.

For CLIP-PGP, the experimental setting is that, on the vision side, we only set a single trainable image prompt shared by all tasks. As for the text side, we follow the operation as (Zhou et al., 2022), we set trainable text prompt for each class, which is only trained at the corresponding task. In comparison with CLIP and CLIP-PGP, we both set the image prompt length as 5, epochs as 5, and batch size as 32 for 10-Split-CIFAR100. Specifically in CLIP-PGP, we set $\epsilon = 0.90$ for extraction of image prompt gradient projection matrix.

H RESULT TABLE WITH THE STANDARD DEVIATION VALUES

We conduct 3 runs of our method and competitors, additional results with the standard deviation values on different datasets are shown in Table 5

I COMPARISON WITH BASELINES AND UPPER-BOUND

We compare the performance of prompt-based methods with and without PGP in Table 6. To be consistent with previous works (Wang et al., 2022c), we report the difference between accuracy performance of the Upper-Bound and the model as a metric. We observe that PGP again sets a new SOTA in this setting. As we compare the Diff performance of DualPrompt and L2P with and without PGP, we again notice an obvious improvement.

Table 5: Class incremental learning on different datasets along with the standard deviation values.

Method	Exemplar	10-Split-CIFAR100		20-Split-CIFAR100		10-Split-ImageNet-R	
		ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)
BiC	5000	81.42±0.85	17.31±1.02	73.02±0.93	6.23±1.17	64.63±1.27	22.25±1.73
DER++	5000	83.94±0.34	14.55±0.73	-	-	66.73±0.87	20.67±1.24
ICaRL	5000	66.00±0.66	5.33±0.94	78.02±0.71	5.80±1.02	-	-
DER+MCG	2000	67.62±0.04	14.64±0.53	65.84±0.18	13.72±1.28	-	-
BiC	1000	66.11±1.76	35.24±1.64	63.12±2.35	21.89±1.93	52.14±1.08	36.70±1.05
DER++	1000	61.06±0.87	39.87±0.99	-	-	55.47±1.31	34.64±1.50
ICaRL	1000	61.25±0.63	14.19±1.14	71.32±0.86	15.98±1.35	-	-
FT	✗	33.61±0.85	86.87±0.20	33.52±0.94	53.69±0.52	28.87±1.36	63.80±1.50
EWC	✗	47.01±0.29	33.27±1.17	36.73±0.57	35.19±1.98	35.00±0.43	56.16±0.88
LWF	✗	60.69±0.63	27.77±2.17	39.12±0.87	57.91±3.06	38.54±1.23	52.37±0.64
L2P	✗	83.77±0.16	6.63±0.05	81.29±0.43	8.96±0.38	60.44±0.41	9.00±0.86
L2P-PGP(Ours)	✗	**84.34±0.08**	**5.59±0.05**	**82.00±0.56**	**8.39±0.62**	**61.40±0.34**	**8.03±0.03**
DualPrompt	✗	86.50±0.45	5.77±0.02	82.98±0.47	8.20±0.08	68.13±0.10	4.68±0.19
DualPrompt-PGP(Ours)	✗	**86.92±0.05**	**5.35±0.19**	**83.74±0.01**	**7.91±0.15**	**69.34±0.05**	**4.53±0.04**
Upper-Bound	-	90.85±0.12	-	90.85±0.12	-	79.13±0.18	-

Table 6: Comparison with baselines in terms of differences between accuracy performance of the Upper-Bound and the model. The Upper-Bound denotes the model performance when trained with access to all tasks at the same time. we use **Diff = Upper-Bound ACC - Method ACC.**

Method	10-Split-CIFAR100		20-Split-CIFAR100		10-Split-ImageNet-R	
	ACC(↑)	Diff(↓)	ACC(↑)	Diff(↓)	ACC(↑)	Diff(↓)
Upper-Bound	90.85	-	90.85	-	79.13	-
L2P	83.77	7.08	81.29	9.56	60.44	18.69
L2P-PGP	**84.34**	**6.51**	**82.00**	**8.85**	**61.40**	**17.73**
DualPrompt	86.50	4.35	82.98	7.87	68.13	11.00
DualPrompt-PGP	**86.92**	**3.93**	**83.74**	**7.11**	**69.34**	**9.79**

J TASK INCREMENTAL SETTING

We compare L2P-PGP with L2P and representative SOTA competitors: EWC (Kirkpatrick et al., 2017), LWF (Li & Hoiem, 2017), A-GEM (Chaudhry et al., 2018), OWM (Zeng et al., 2019), Adam-NSCL (Wang et al., 2021), Connector (Lin et al., 2022a), results as shown in Table 7.

Both on 10-Split-CIFAR100 and 20-Split-CIFAR100 datasets, although L2P has already achieved higher accuracy and lower forgetting compared with other CNN methods, our method further improves its accuracy and reduces its forgetting with the aid of prompt gradient projection and L2P-PGP achieves new SOTA performance. On 10-Split-CIFAR100 dataset, PGP improves L2P by 0.10 on accuracy, 0.05 on forgetting, and on 20-Split-CIFAR100, PGP improves L2P by 0.11 on accuracy, 0.11 on forgetting.

Table 7: Task incremental learning results on different datasets.

Method	10-Split-CIFAR100		20-Split-CIFAR100	
	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)
EWC	70.77	2.83	71.66	3.72
LWF	70.70	6.27	74.38	9.11
A-GEM	49.57	1.13	61.91	6.88
OWM	68.89	1.88	68.47	3.37
Adam-NSCL	73.77	1.60	75.95	3.66
Connector	79.79	0.92	80.80	5.00
L2P	97.43	0.22	98.47	0.39
L2P-PGP	**97.53**	**0.17**	**98.58**	**0.28**

K CONTINUAL LEARNING RESULTS ON MULTI-MODEL BACKBONE, COMPARISON BETWEEN CLIP-PGP WITH CLIP

We conduct our experiments on 10-Split-CIFAR100 dataset under class incremental setting and task incremental setting respectively, as shown in Table 8. Results show that, our method has improved the performance a lot for both the above settings, proving that our method is also useful in the vision-language models, which further enlarges the scope of our method.

Table 8: Comparison to *CLIP* model **with/without** gradient projection method on 10-Split-CIFAR100 with class/task incremental settings.

Settings	**Class Incremental**		**Task Incremental**	
Models	Accuracy	Forgetting	Accuracy	Forgetting
CLIP	73.76	5.60	92.69	2.34
CLIP-PGP(Ours)	**79.47**(+5.71)	**4.23**(-1.37)	**93.00**(+0.31)	**1.58**(-0.76)

L CLASS INCREMENTAL LEARNING RESULTS ON DIFFERENT BACKBONES, COMPARISON BETWEEN OURS WITH BASELINES

To show the efficacy of proposed method on different pre-trained backbones, we evaluate our method by extending two distinct pre-trained models, namely ViT-DINO and ViT-SAM (Caron et al., 2021; Chen et al., 2021). The results are shown in the Table 9. Additionally, we tested our method on 10-Split-CIFAR100 and 5-Split-CUB200 dataset based on three pre-trained ViTs: ImageNet-21K, DINO, and SAM, further validating the effectiveness of our method on non-ImageNet datasets (Wah et al., 2011; Krizhevsky et al., 2009).

Table 9: Comparison to distinct pre-trained backbones between baselines and ours. **Red** parts show significant improvements ($>$**1**).

		10-Split-CIFAR100		**5-Split-CUB200**	
Method	**Pretrained-Dataset**	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)
L2P	ImageNet-21K	83.77	6.63	74.88	5.39
L2P-PGP	**ImageNet-21K**	**84.34**(+0.57)	**5.59**(-1.04)	**75.15**(+0.27)	**4.51**(-0.88)
DualPrompt	ImageNet-21K	86.50	5.77	82.02	4.23
DualPrompt-PGP	**ImageNet-21K**	**86.92**(+0.42)	**5.35**(-0.42)	**82.46**(+0.44)	**3.76**(-0.47)
L2P	SAM	83.93	6.68	73.98	6.77
L2P-PGP	**SAM**	**84.26**(+0.33)	**5.64**(-1.04)	**76.45**(+2.47)	**5.91**(-0.86)
DualPrompt	SAM	86.11	6.08	82.02	4.73
DualPrompt-PGP	**SAM**	**86.92**(+0.81)	**5.04**(-1.04)	**82.28**(+0.26)	**4.65**(-0.08)
L2P	DINO	67.35	9.69	44.10	9.77
L2P-PGP	**DINO**	**70.60**(+3.25)	**4.73**(-4.96)	**44.80**(+0.70)	**6.06**(-3.71)
DualPrompt	DINO	64.18	23.87	50.88	10.10
DualPrompt-PGP	**DINO**	**73.33**(+9.15)	**10.27**(-13.60)	**51.03**(+0.15)	**9.06**(-1.04)

M PGP WITH PROMPT NUMBER AND PROMPT WIDTH

In this section, for L2P-PGP model, we set distinct parameters in prompt numbers and prompt widths on 10-Split-CIFAR100 dataset, and further validate the efficiency of prompt gradient projection method. Results are shown in Table 10. In our setting, we set a single prompt mode, that all tasks share a single prompt for training. We think, in this way, we can deeply uncover the potential of our method and avoid interference caused by choosing prompts. Results show that, models with prompt gradient projection, all have higher accuracy and lower forgetting than those without, which proves that our method could be effective in distinct prompt numbers and widths, even with a hard single prompt setting.

Table 10: Comparison L2P with L2P-PGP on 10-Split-CIFAR100 dataset. Width and number mean prompt width and prompt number respectively.

	L2P		L2P-PGP	
Width	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
5	82.64	6.73	82.77	6.58
10	82.09	7.07	82.16	6.74
15	83.09	6.38	84.21	5.62
20	83.42	6.38	83.87	5.89
25	83.69	6.49	83.85	6.39
30	83.87	6.46	84.39	6.44

	L2P		L2P-PGP	
Number	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
1	82.64	6.73	82.77	6.58
3	84.17	5.92	84.19	5.60
5	83.23	6.66	83.82	6.62
7	83.87	7.13	84.44	6.58
9	84.11	6.60	84.15	6.52

N PGP WITH PREFIX WIDTH AND PREFIX PREPENDED LAYER

In this section, for DualPrompt-PGP model, we discuss whether prompt gradient projection could be efficient in different prefix widths and prepended layers. As the setting in Appendix M, we choose a single prefix mode based on the same reason. We conduct experiments on 10-Split-CIFAR100 and 10-Split-TinyImageNet. Final results are shown in Table 11 and Table 12. We also show some cases with curves of accuracy and forgetting metrics changing in all tasks, as in Figure 6 and Figure 7.

Table 11: Comparison DualPrompt with DualPrompt-PGP on 10-Split-CIFAR100 dataset. Width and layer mean prefix width and prefix prepended layer index respectively.

	DualPrompt		DualPrompt-PGP	
Width	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
5	81.08	7.64	81.49	7.08
6	81.32	7.12	81.70	6.89
7	81.67	7.51	81.95	6.77
8	81.67	7.48	81.92	7.06
9	81.74	6.49	81.88	6.21
10	81.58	6.93	81.63	6.78

	DualPrompt		DualPrompt-PGP	
Layer	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
0	81.08	7.64	81.49	7.08
0,1	82.22	5.78	82.75	5.67
0,1,2	83.85	5.62	84.69	4.38
0,1,2,3	84.55	5.03	84.58	4.84
0,1,2,3,4	84.59	5.60	84.74	5.04

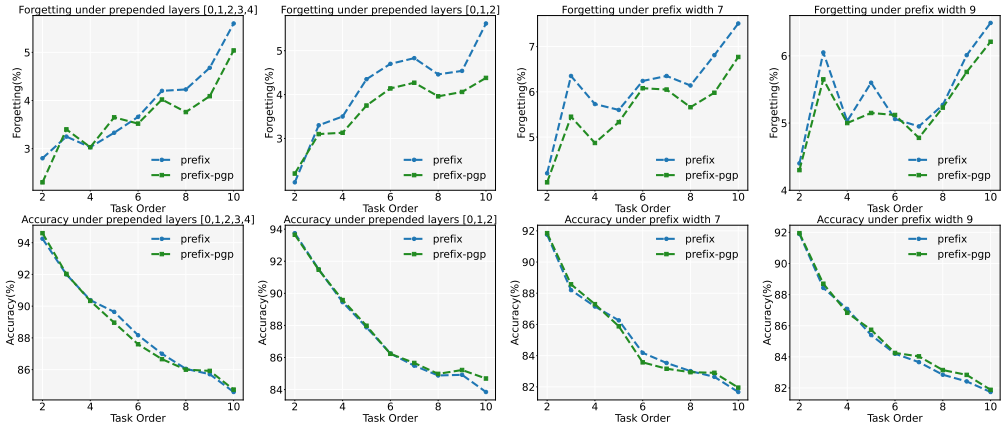

Figure 6: Changing curves of accuracy and forgetting metrics with different prepended layers and prefix widths on 10-Split-CIFAR100 dataset.

Results are similar to the discussion in Appendix M. Whether on 10-Split-CIFAR100 or 10-Split-TinyImageNet, models with prompt gradient projection always have better accuracy and lower forgetting than those without. We think it proves that our method can be effective in distinct prefix widths and prepended layers. Notice that we name the baseline as "prefix" and our method as "prefix-pgp".

Table 12: Comparison DualPrompt with DualPrompt-PGP in different settings on 10-Split-TinyImageNet dataset. Width and layer mean prefix width and prefix prepended layer index respectively.

	DualPrompt		DualPrompt-PGP	
Width	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
5	81.58	4.63	81.79	4.51
6	81.39	4.66	81.67	4.50
7	81.60	4.93	81.78	4.43
8	81.36	4.63	81.65	4.44
9	81.55	4.80	81.93	4.70
10	82.20	4.34	82.22	3.96

	DualPrompt		DualPrompt-PGP	
Layer	ACC(↑)	FOR(↓)	ACC(↑)	FOR(↓)
0	81.58	4.63	81.79	4.51
0,1	82.98	4.29	83.33	3.98
0,1,2	83.66	4.11	83.76	3.96
0,1,2,3	83.64	4.62	84.51	3.72
0,1,2,3,4	83.61	4.68	83.95	4.23

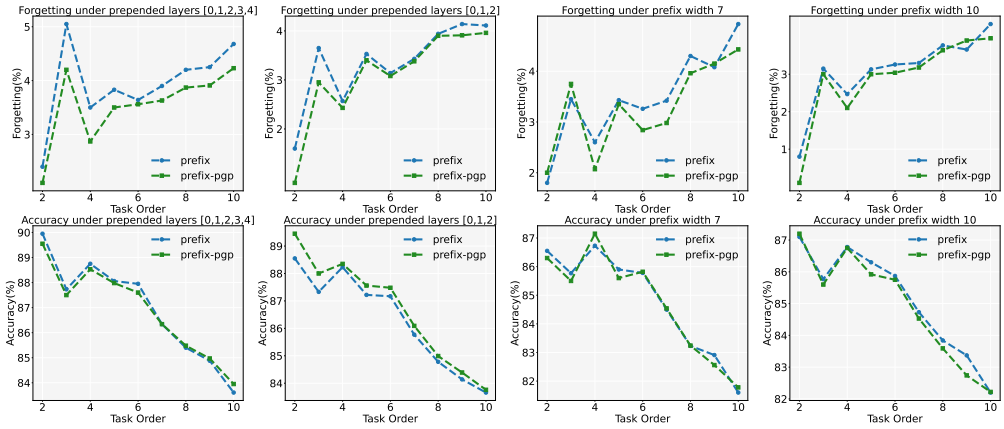

Figure 7: Changing curves of accuracy and forgetting metrics with different prepended layers and prefix widths on 10-Split-TinyImageNet dataset.

O T-SNE VISUALIZATION

To better visualize the improvement of our method, we choose L2P and L2P-PGP models. Training on 10-Split-CIFAR100 dataset, we show the T-SNE results of samples from task 1 across models in various tasks. We pick up logits processed by classifier to report.

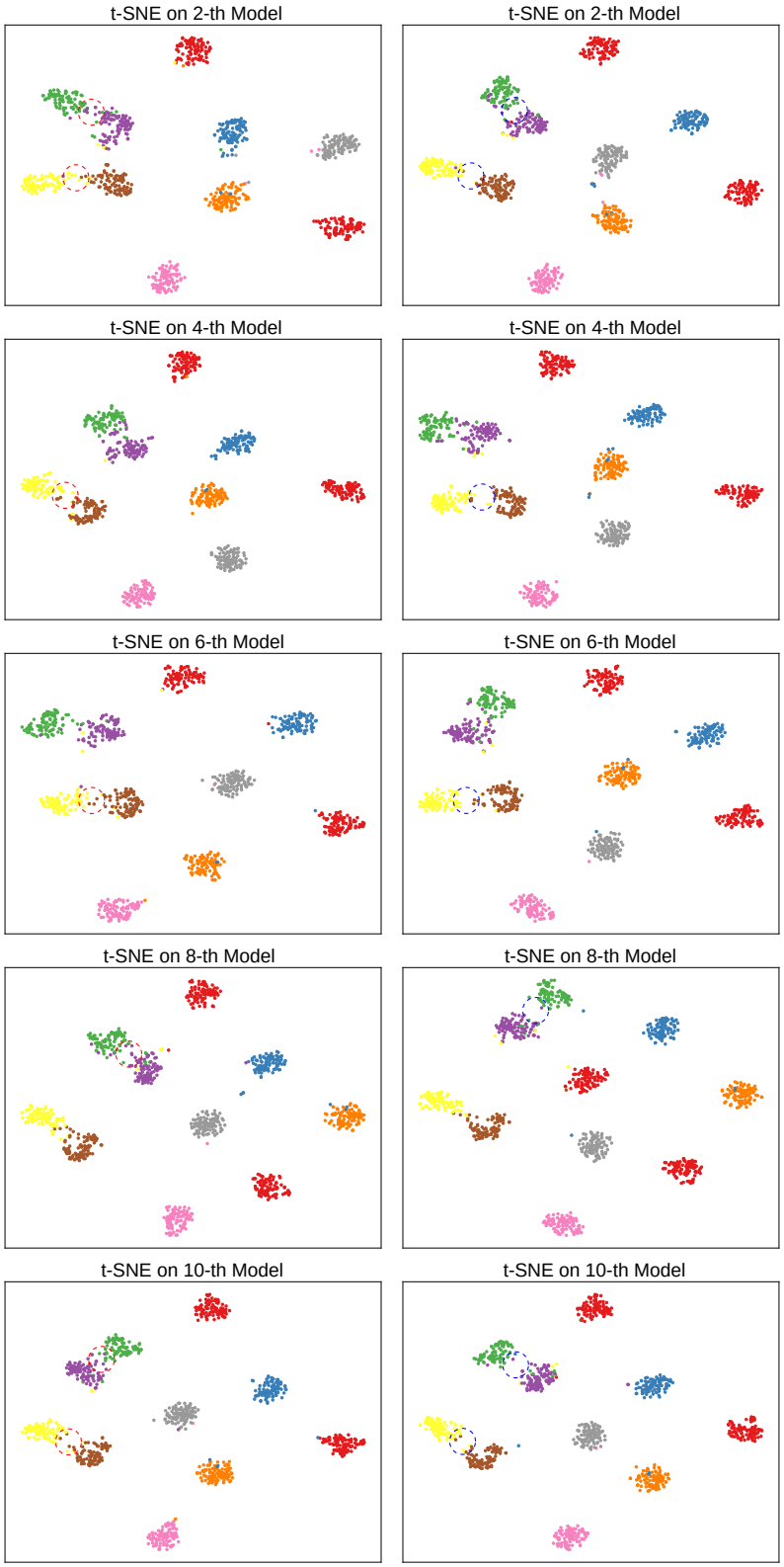

Figure 8: T-SNE results of L2P and L2P-PGP on 10-Split-CIFAR100 dataset. The left column represents L2P, and the right column represents L2P-PGP. The red circle means the drawback existing in L2P, and the blue circle shows the improvement of our method.

P ALGORITHM

Algorithm 1: Prompt Gradient Projection For L2P (Training phase)

Input: Pre-trained ViT model f_θ, embedding layer ϕ_θ, classifier head f_c, number of tasks T, training set $\{\{X_i^t, y_i^t\}_{i=1}^{n_t}\}_{t=1}^{T}$, sampling set $\{\{X_{si}^t, y_{si}^t\}_{i=1}^{n_{st}}\}_{t=1}^{T}$, prompt pool $\{p_j\}_{j=1}^{M}$, projection matrix $V_{t,0}$, number of training epochs E, learning rate η, loss function $\mathcal{L}_x$

Output: prompt pool $\{p_j\}_{j=1}^{M}$, classifier head f_c

initialize: f_c, $\{p_j\}_{j=1}^{M}$.

for t = *1, ..., T* **do**
  for e = *1, ..., E* **do**
    Draw a mini-batch $B = \{(X_i^t, y_i^t)\}_{i=1}^{n_t}$.
    for (X, y) *in* B **do**
      Embed X into sequence x_t by $x_t = \phi_\theta(X)$.
      Select prompt p_x from $\{p_j\}_{j=1}^{M}$.
      Prepend x_t with p_x by $x_p = [p_x; x_t]$.
      Obtain prediction by $\hat{y} = f_c(f_\theta(x_p))$.
    end
    Calculate per batch loss $\mathcal{L}_B$ by accumulating $\mathcal{L}_x(y, \hat{y})$.
    # Gradient projection
    if *t = 1* **then**
      Update p by $p \leftarrow p - \eta \nabla_p \mathcal{L}_B$.
    else
      Update p by $p \leftarrow p - \eta \nabla_p \mathcal{L}_B V_{t,0} V_{t,0}^T$.
    end
  end
  # Gradient projection matrix update
  Initialize the sets of sampled embedding sequences and prompts: $X_t = \{\}$, $P_t = \{\}$.
  for (X_{si}^t, y_{si}^t) *in* $\{(X_{si}^t, y_{si}^t)\}_{i=1}^{n_{st}}$ **do**
    Sample set of embedding sequences X_t by concatenation of X_t and $\phi_\theta(X_{si}^t)$.
  end
  for p *in* $\{p_j\}_{j=1}^{M}$ *and* $p \in p_x$ **do**
    Sample set of prompts P_t by concatenation of P_t and p.
  end
  Update $V_{t,0}$ by X_t and P_t according to Appendix C.
end

Algorithm 2: Prompt Gradient Projection For L2P (Testing phase)

Input: Pre-trained ViT model f_θ, embedding layer ϕ_θ, classifier head f_c, number of tasks T, test set $\{\{X_i^t\}_{i=1}^{n_t}\}_{t=1}^{T}$, prompt pool $\{p_j\}_{j=1}^{M}$

Output: prediction $\hat{y}$

for $t = 1, ..., T$ **do**
  for X_i^t *in* $\{X_i^t\}_{i=1}^{n_t}$ **do**
    Embed X_i^t into sequence x_t by $x_t = \phi_\theta(X_i^t)$.
    Select prompt p_x from $\{p_j\}_{j=1}^{M}$.
    Prepend x_t with p_x by $x_p = [p_x; x_t]$.
    Obtain prediction by $\hat{y} = f_c(f_\theta(x_p))$.
  end
end