Prompt Gradient Projection For Continual Learning

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

East China Normal University

ICLR 2024 Spotlight
2024-04-01
https://jingyangqiao.github.io/

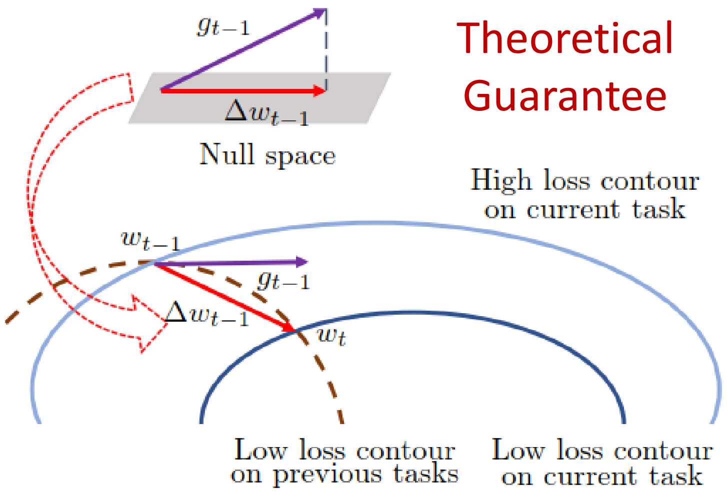
Prompt Gradient Projection For Continual Learning

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

ICLR 2024

Motivation

Gradient Projection Method



Theoretical Guarantee

$$W_t^l x_t^l = (W_{t-1}^l + \Delta W) x_t^l$$
$$= W_{t-1}^l x_t^l + \Delta W x_t^l$$

if.

$$\Delta W x_t^l = 0$$

thus.

$$\boldsymbol{W_t^l x_t^l = W_{t-1}^l x_t^l}$$

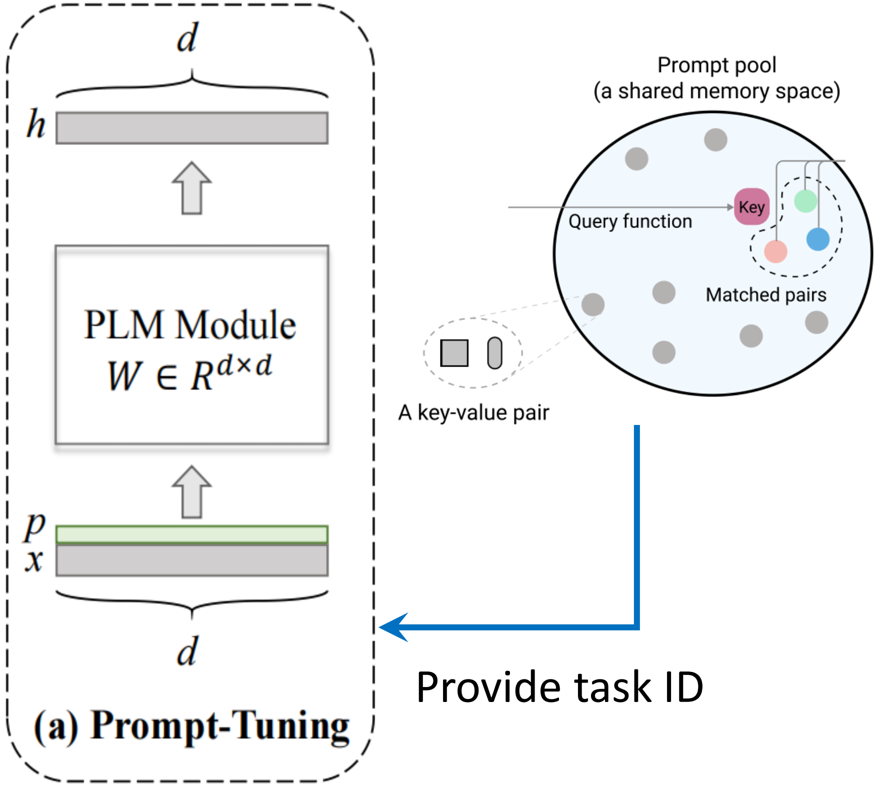
Theoretical Support?

Drawbacks:

(1) Only applicable for CNN architecture

(2) Inference needs task ID

Prompt Tuning & Key-Query Mechanism



[1] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. arXiv preprint *arXiv*:2103.09762, 2021.
[2] Shipeng Wang, Xiaorong Li, Jian Sun, et al. Training networks in null space of feature covariance for continual learning. *CVPR*, pp. 184–193, 2021.
[3] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, et al. Learning to prompt for continual learning. *CVPR*, pp. 139–149, 2022c.

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

Proposition

From the perspective that the **old inputs** from previous tasks have **the same outputs** after **learning a new task**:

proposition 1 *To better preserve old knowledge, the update of network would satisfy the following equation:*

$$f_\theta(p_{t+1}, x_t) = f_\theta(p_t, x_t), \tag{2}$$

where x_t denotes the feature embeddings from old task t, p_t and p_{t+1} denote the prompts trained at task t and $t+1$.

The ideal condition of anti-forgetting in prompt-tuning continual learning method!

Prompt Gradient Projection For Continual Learning

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie (✉)

ICLR 2024

Deduction

(1) From the perspective of self-attention mechanism, we have:

$$A_t^{t+1} = softmax(\frac{Q_t^{t+1} {K_t^{t+1}}^T}{\sqrt{(\frac{d}{h})}}).$$

constant

$$\begin{cases} (Q_t^{t+1} = W_q Z_t^{t+1}) \\ (K_t^{t+1} = W_k Z_t^{t+1}) \end{cases} \qquad Z_t^{t+1} = \begin{bmatrix} p_{t+1} \\ x_t \end{bmatrix}$$

$$Q_t^{t+1} {K_t^{t+1}}^T = W_q \boxed{Z_t^{t+1} {Z_t^{t+1}}^T} W_k^T$$

Changed part during training

W_q and W_k, the weights of visual encoder, are frozen and unchanged during training.

$$Z_t^{t+1} \cdot {Z_t^{t+1}}^T = \begin{bmatrix} p_{t+1} \\ x_t \end{bmatrix} \begin{bmatrix} p_{t+1}^T & x_t^T \end{bmatrix} = \begin{bmatrix} p_{t+1} p_{t+1}^T & p_{t+1} x_t^T \\ x_t p_{t+1}^T & x_t x_t^T \end{bmatrix}$$

Old input with new prompt

(2) Similarly, we have:

$$Z_t^t \cdot {Z_t^t}^T = \begin{bmatrix} p_t \\ x_t \end{bmatrix} \begin{bmatrix} p_t^T & x_t^T \end{bmatrix} = \begin{bmatrix} p_t p_t^T & p_t x_t^T \\ x_t p_t^T & x_t x_t^T \end{bmatrix}$$

Old input with old prompt

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

Deduction

(3) In order to achieve Eq.(2), the corresponding item should be equal:

$$\boxed{\begin{cases} p_{t+1}p_{t+1}^T = p_t p_t^T, \\ x_t p_{t+1}^T = x_t p_t^T, \\ p_{t+1}x_t^T = p_t x_t^T. \end{cases}}$$

we divide p_{t+1} into p_t and Δp, where Δp is the gradient of prompts when training task $t+1$.

(4) For the **first term**, we have:

$$p_{t+1}p_{t+1}^T = (p_t + \Delta p)(p_t + \Delta p)^T = p_t p_t^T + p_t \Delta p^T + \Delta p p_t^T + \Delta p \Delta p^T.$$

Here we ignore the high-order infinitesimal term of $\Delta p \Delta p^T$

Thus, if we have: $p_t \Delta p^T = 0$

Then, we have: $p_{t+1}p_{t+1}^T = p_t p_t^T$

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

Deduction

(5) In the same way, the **second condition** can be transformed to:

$$x_t p_{t+1}^T = x_t(p_t^T + \Delta p^T) = x_t p_t^T + x_t \Delta p^T = x_t p_t^T$$

Thus, if we have: $x_t \Delta p^T = 0$

Then, we have: $x_t p_{t+1}^T = x_t p_t^T$

(6) In summary, with the satisfied the following equations, we can achieve Eq.(2), the ***proposition***.

$$\boxed{\begin{cases} x_t \Delta p^T = 0, \\ p_t \Delta p^T = 0. \end{cases}}$$

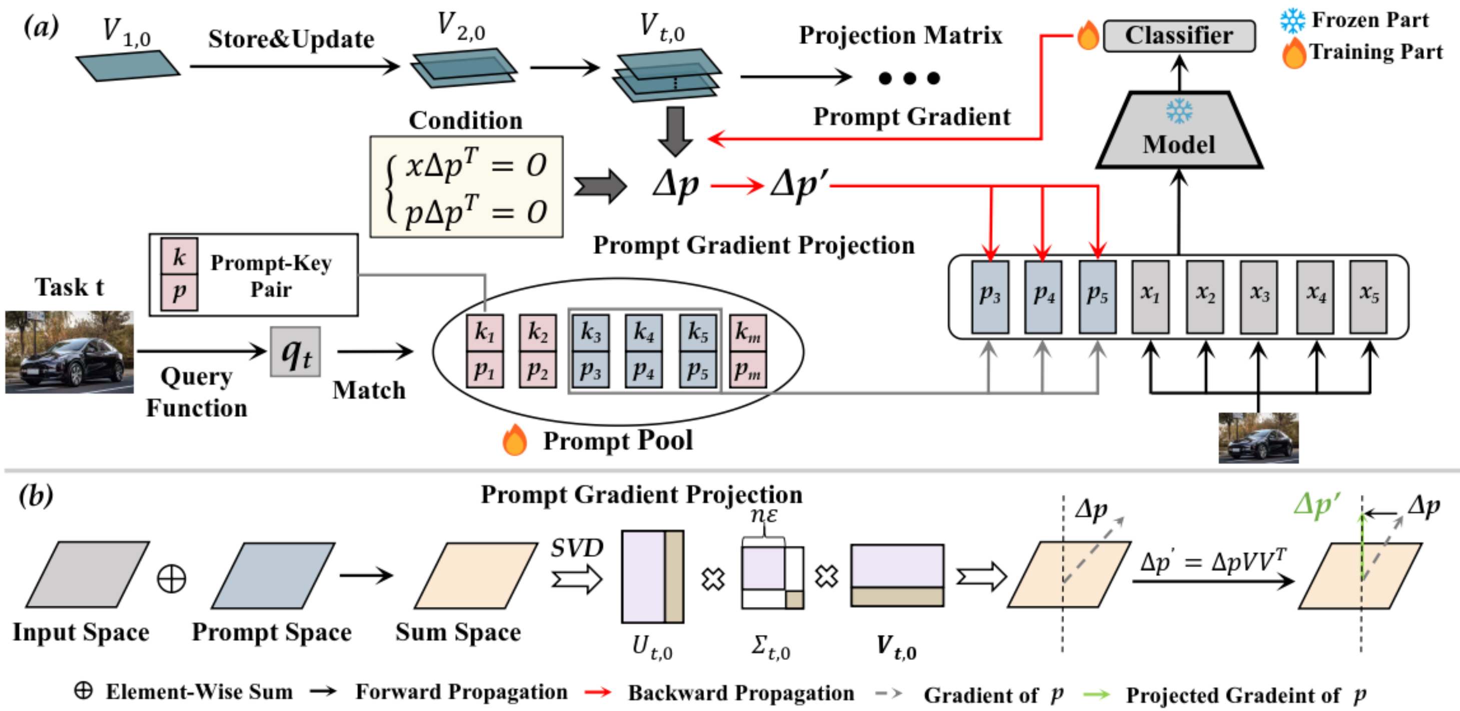
Most important euations !

Prompt Gradient Projection For Continual Learning

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie (✉)

ICLR 2024

Method



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

Proof.

Realize the most important equatioins

(1) $$x_t = U_t \Sigma_t V_t^T$$

(2) $$\begin{cases} \Sigma_t = \begin{bmatrix} \Sigma_{t,1} & O \\ O & \Sigma_{t,0} \end{bmatrix} \\ V_t = [V_{t,1}, V_{t,0}] \end{cases}$$

(3) $$x_t[V_{t,1}, V_{t,0}] = U_t \begin{bmatrix} \Sigma_{t,1} & O \\ O & \Sigma_{t,0} \end{bmatrix} \rightarrow x_t V_{t,0} = U_t \begin{bmatrix} O \\ \Sigma_{t,0} \end{bmatrix} \approx O$$

(4) $$\boxed{x_t \Delta p^T = x_t (\Delta p V_{t,0} V_{t,0}^T)^T = x_t V_{t,0} V_{t,0}^T \Delta p^T = O}$$

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie (✉)

Experiment Settings

Datasets: We evaluate our method on **1) 10/20-Split-CIFAR100** (Krizhevsky et al., 2009), constructed by splitting the 100 classes into 10 tasks/20 tasks. **2) 10-Split-TinyImageNet** (Abai & Rajmalwar, 2019), constructed by splitting the 200 classes into 10 tasks. **3) 10-Split-ImageNet-R** (Hendrycks et al., 2021), constructed by splitting the 200 classes into 10 tasks.

Implementation: We use L2P (Wang et al., 2022c), DualPrompt (Wang et al., 2022b), and CLIP (Radford et al., 2021) as our baselines, with prompt gradient projection for updating. We follow their original settings, and the only difference is we train DualPrompt with extra 15 epochs on CIFAR100 suggested by (Khan et al., 2023). Detailed experiment information could be seen in Appendix G.

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 $\beta_2 = 0.999$.

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

Experiment Settings

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.

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

Experiment Results - Class Incremental Learning (ViT)

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

Method	Exemplar	10-Split-CIFAR100 ACC(↑)	10-Split-CIFAR100 Forgetting(↓)	20-Split-CIFAR100 ACC(↑)	20-Split-CIFAR100 Forgetting(↓)	10-Split-ImageNet-R ACC(↑)	10-Split-ImageNet-R 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	-

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie

Experiment Results - Class Incremental Learning (distinct pre-trained ViTs)

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

Method	Pretrained-Dataset	10-Split-CIFAR100		5-Split-CUB200	
		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)**

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie(✉)

Experiment Results - Class/Task Incremental Learning (CLIP)

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)

Jingyang Qiao*, Zhizhong Zhang*, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie

Experiment Results - Online Class Incremental Learning

Table 3: Main results of online class incremental learning in terms of accuracy and forgetting. The comparison is made between our approach and the corresponding baselines.

Method	10-Split-CIFAR100		20-Split-CIFAR100		10-Split-TinyImageNet	
	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)	ACC(↑)	Forgetting(↓)
L2P	79.99	8.19	77.63	11.33	78.69	5.83
L2P-PGP	**80.29**	**7.73**	**78.34**	**9.33**	**79.47**	**5.19**
DualPrompt	80.93	5.51	79.02	6.89	82.20	3.62
DualPrompt-PGP	**81.02**	**5.41**	**79.41**	**6.75**	**82.57**	**3.57**

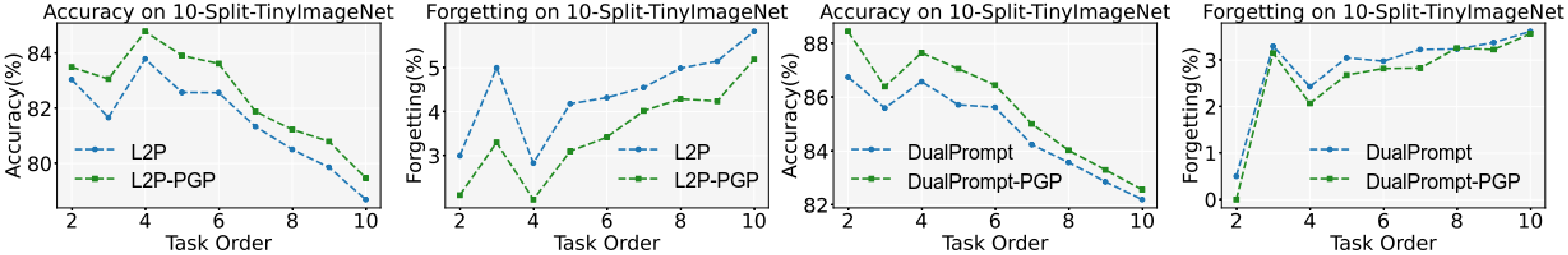


Figure 3: Task-by-task performance changing curves in terms of accuracy and forgetting under online class incremental setting.