Exploring Example Influence in Continual Learning (Appendix)

Qing Sun*, **Fan Lyu***, **Fanhua Shang**, **Wei Feng**, **Liang Wan**†
College of Intelligence and Computing, Tianjin University
{sssunqing, fanlyu, fhshang, wfeng, lwan}@tju.edu.cn
https://github.com/SSSunQing/Example_Influence_CL

A Notation

We list the mentioned notation in Table 1 for quickly looking up.

Table 1: Notation related to the paper.

Symbol	Description	Symbol	Description
$\boldsymbol{\theta}_t$	The model trained after the t-th task	P_t	$p(\mathcal{D}_t^{\text{tst}}\|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_t^{\text{tst}}\|\boldsymbol{\theta}_{t-1})$, Plasticity
α	Optimization step size	$\mathbf{H}$	$\nabla^2_{\hat{\boldsymbol{\theta}}}\ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}})$; Hessian assumed positive define
$\mathcal{D}_t$	$\{(x_t^{(n)}, y_t^{(n)})\}_{n=1}^{N_t}$; The t-th dataset	ϵ	Example-level perturbation
$\mathcal{D}_t^{\text{trn}}$	The training set of the t-th dataset	$\mathbf{E}$	Batch-level perturbation
$\mathcal{D}_t^{\text{tst}}$	The testing set of the t-th dataset	$\hat{\theta}_{\epsilon,\mathcal{B}}$	Pseudo updated model with example-level perturbation ϵ
$\mathcal{M}$	The memory buffer sampled from training set	$\hat{\theta}_{\mathbf{E},\mathcal{B}}$	Pseudo updated model with batch-level perturbation $\mathbf{E}$
$\mathcal{B}_{\text{old}}$	A mini-batch sampled from $\mathcal{M}$	$\mathbf{I}(\mathcal{V}_{\text{old}}, \mathcal{B})$	Influence on S from mini-batch $\mathcal{B}$
$\mathcal{B}_{\text{new}}$	A mini-batch sampled from $\mathcal{D}_t^{\text{trn}}$	$\mathbf{I}(\mathcal{V}_{\text{new}}, \mathcal{B})$	Influence on P from mini-batch $\mathcal{B}$
$\mathcal{V}_{\text{old}}$	Validation set of old tasks	γ	Weight fusion factor
$\mathcal{V}_{\text{new}}$	Validation set of new task	$\mathbf{I}^*$	Fusion Influence on SP from mini-batch $\mathcal{B}$
S_t^k	$p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_{t-1}, \mathcal{D}_t^{\text{trn}}) - p(\mathcal{D}_k^{\text{tst}}\|\boldsymbol{\theta}_k), \quad k < t$ Stability	l	Loss for an example or average loss for a mini-batch
$\mathbf{L}$	Loss vector for a mini-batch		

B Influence Function

In this section, we illustrate how to get the Influence Function. Given a parameter $\boldsymbol{\theta}$ up to update, $\hat{\boldsymbol{\theta}}$ is the updated parameter using the training data, *i.e.*, $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell(\mathcal{B}, \boldsymbol{\theta})$. First, we set a small weight to a specific example x

$$\hat{\boldsymbol{\theta}}_{\epsilon,x} \coloneqq \arg\min_{\boldsymbol{\theta}} \ell(\mathcal{B}, \boldsymbol{\theta}) + \epsilon\ell(x, \boldsymbol{\theta}), \quad x \in \mathcal{B},$$

where ϵ is a small weight. Then, the variation of parameter can be shown as the IF on parameters

$$I(x, \hat{\boldsymbol{\theta}}) = \left.\frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\right|_{\epsilon=0} = -\mathbf{H}_{\hat{\boldsymbol{\theta}}}^{-1} \nabla_{\boldsymbol{\theta}} \ell(x, \hat{\boldsymbol{\theta}}).$$

Last, the influence from one training example to a testing example can be obtained by the chain rule

$$I(x^{\text{trn}}, x^{\text{tst}}) = \frac{\partial \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \left.\frac{\partial \hat{\boldsymbol{\theta}}_{\epsilon,x}}{\partial \epsilon}\right|_{\epsilon=0} = -\nabla_{\boldsymbol{\theta}} \ell(x^{\text{tst}}, \hat{\boldsymbol{\theta}}) \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} \ell(x^{\text{trn}}, \hat{\boldsymbol{\theta}}),$$

where $\mathbf{H} = \nabla^2_{\hat{\boldsymbol{\theta}}} \ell(\mathcal{B}, \hat{\boldsymbol{\theta}})$ is a Hessian and is assumed as positive definite.

*Co-first authors.
†Corresponding author.

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

Table 2: Comparisons on Split CIFAR-10, averaged across 5 runs. Red and blue values mean the best in our methods and the compared methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	Split CIFAR-10 (Class-Increment)							
Fine-tune	19.66±0.04							
Joint	91.79±0.68							
Buffer size	$\|\mathcal{M}\| = 300$				$\|\mathcal{M}\| = 500$			
	A_1	A_∞	A_{m}	BWT	A_1	A_∞	A_{m}	BWT
GDUMB	-	36.92±1.86	-	-	-	44.27±0.41	-	-
GEM	93.90±0.55	37.51±2.06	55.43±0.31	-70.48	92.76±1.13	36.95±2.34	57.36±1.02	-69.76
AGEM	96.57±0.40	20.02±0.23	45.57±0.40	-95.69	96.56±0.11	20.01±0.16	46.52±1.04	-95.70
HAL	91.30±1.95	24.45±2.09	46.34±1.77	-83.57	91.96±0.64	27.94±2.15	49.05±1.00	-80.02
MIR	96.70±0.12	38.53±1.72	56.96±1.16	-72.72	96.65±0.10	42.65±1.46	59.99±0.69	-67.50
GSS	96.53±0.20	35.89±2.46	54.33±1.35	-75.80	96.55±0.27	41.96±1.08	58.16±0.46	-68.25
GMED	96.65±0.24	38.12±0.99	58.92±0.67	-73.17	96.65±0.26	43.68±1.74	62.56±0.56	-66.22
ER	96.73±0.35	34.19±1.35	53.72±0.39	-78.18	96.74±0.08	40.45±2.14	57.69±1.41	-70.36
Ours	**96.87±0.09**	42.42±1.94	63.52±0.70	-68.07	**96.82±0.21**	49.16±1.48	67.88±0.82	-59.58
Ours+RehSel	96.85±0.09	**43.76±0.52**	**63.69±0.62**	**-66.37**	96.81±0.19	**50.10±1.32**	**68.28±0.88**	**-58.39**

Method	Split CIFAR-10 (Task-Increment)							
Fine-tune	65.27±2.28							
Joint	98.16±0.09							
Buffer size	$\|\mathcal{M}\| = 300$				$\|\mathcal{M}\| = 500$			
	A_1	A_∞	A_{m}	BWT	A_1	A_∞	A_{m}	BWT
GDUMB	-	73.22±0.67	-	-	-	78.06±1.41	-	-
GEM	96.62±0.05	89.34±0.99	92.49±0.36	-9.10	96.73±0.25	90.42±1.23	92.93±0.27	-7.89
AGEM	96.78±0.29	85.52±1.01	90.16±0.17	-14.07	96.71±0.10	86.45±1.06	90.90±0.70	-12.83
HAL	91.41±1.86	79.90±2.25	83.78±1.62	-14.39	92.03±0.64	81.84±2.13	84.19±1.46	-12.73
MIR	96.76±0.09	88.50±1.30	90.87±0.50	-10.33	96.73±0.08	90.63±0.63	91.99±0.46	-7.62
GSS	96.56±0.18	88.05±1.52	90.60±0.82	-10.63	96.57±0.27	90.38±0.87	92.19±0.59	-7.74
GMED	96.73±0.24	88.91±1.16	91.20±0.60	-9.77	96.72±0.22	89.72±1.25	92.10±0.65	-8.75
ER	96.93±0.07	88.97±0.67	91.12±0.79	-9.95	96.79±0.08	90.60±0.74	92.28±0.32	-7.75
Ours	97.10±0.12	89.40±0.93	92.54±0.38	-9.63	**97.31±0.18**	90.91±0.60	**93.38±0.31**	-7.99
Ours+RehSel	**97.11±0.11**	**89.91±0.55**	**92.66±0.23**	**-9.00**	97.30±0.04	**91.41±0.60**	93.28±0.32	**-7.36**

Based on the rehearsal method, we consider the derivative of the loss of a validation set $\mathcal{V}$ of a mini-batch $\mathcal{B} = [x_1, x_2, \cdots, x_n]$ to weight vector $\mathbf{E} = [\epsilon_1, \epsilon_2, \cdots, \epsilon_n]^\top$ as $\mathbf{L} = \begin{bmatrix}\ell(x_1, \hat{\boldsymbol{\theta}}) & \ell(x_2, \hat{\boldsymbol{\theta}}) & \cdots & \ell(x_n, \hat{\boldsymbol{\theta}})\end{bmatrix}^\top$. With the Taylor expansion, we have

$$\frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \ell(x_i, \hat{\boldsymbol{\theta}}_\epsilon) + \mathbf{w}^\top \frac{\partial \mathbf{L}}{\partial \boldsymbol{\theta}} = \mathbf{0}.$$

The batch-level influence **vector** can be computed by

$$\mathbf{I}(\mathcal{V}, \mathcal{B}) = \frac{\partial l(\mathcal{V}, \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}})}{\partial \mathbf{E}}\bigg|_{\mathbf{E}=\mathbf{0}} = \frac{\partial l(\mathcal{V}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}}{\partial \mathbf{E}}\bigg|_{\mathbf{E}=\mathbf{0}} = -\nabla_{\boldsymbol{\theta}} l(\mathcal{V}, \hat{\boldsymbol{\theta}}) \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}}^\top \mathbf{L}(\mathcal{B}, \hat{\boldsymbol{\theta}}),$$

where

$$\mathbf{H} = \frac{1}{|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}}^2 \ell(x_i, \hat{\boldsymbol{\theta}}_{\epsilon_i}), \quad \frac{\partial \hat{\boldsymbol{\theta}}_{\mathbf{E},\mathcal{B}}}{\partial \mathbf{E}}\bigg|_{\mathbf{E}=\mathbf{0}} = -\mathbf{H}^{-1} \left[\frac{\partial \mathbf{L}}{\partial \boldsymbol{\theta}}\right]^\top.$$

C Comparison Results with std.

In Table 2, 3, and 4, we show more comparison results about the proposed MetaSP with other SOTA methods. For each dataset, we evaluate all methods with task-incremental and class-incremental learning, where the task-incremental setting will offer the task id at the interference while the class-incremental will not. Every method will be evaluated with two different buffer sizes. All the experiments are implemented with 5 fixed seeds from 1231 to 1235 to keep the fair comparison as other CL methods.

Table 3: Comparisons on Split CIFAR-100, averaged across 5 runs. Red and blue values mean the best in our methods and the compared methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	Split CIFAR-100 (Class-Increment)							
Fine-tune	9.14±0.18							
Joint	71.25±0.12							
Buffer size	$\|\mathcal{M}\| = 500$				$\|\mathcal{M}\| = 1000$			
	A_1	A_∞	A_{m}	BWT	A_1	A_∞	A_{m}	BWT
GDUMB	-	11.11±0.60	-	-	-	15.75±0.21	-	-
GEM	85.28±2.26	15.91±0.42	29.38±1.67	-77.07	84.28±2.34	22.79±0.31	34.09±1.75	-68.32
AGEM	85.97±1.27	9.31±0.13	24.60±0.90	-85.18	85.66±1.84	9.27±0.12	24.67±1.07	-84.88
HAL	67.33±1.89	8.20±0.84	22.72±0.71	-65.70	68.06±2.95	10.59±0.78	24.74±1.26	-63.86
MIR	87.38±1.30	13.49±0.18	28.88±1.57	-82.10	87.39±1.24	17.56±0.56	32.48±1.50	-77.60
GSS	86.03±1.91	14.01±0.50	28.00±2.00	-80.03	86.31±1.84	17.87±0.29	31.82±1.86	-76.04
GMED	87.18±1.45	14.56±0.24	33.41±1.37	-80.69	87.29±1.63	18.67±0.30	38.69±1.63	-76.24
ER	87.23±1.65	13.75±0.39	28.88±1.71	-81.65	87.33±1.51	17.56±0.35	32.45±1.78	-77.52
Ours	**88.13±0.80**	18.96±0.40	38.62±0.88	-76.85	**87.58±0.75**	24.78±0.68	45.20±0.97	-69.77
Ours+RehSel	87.81±0.87	**19.28±0.54**	**39.23±0.62**	**-76.14**	87.55±0.68	**25.72±0.48**	**45.48±0.76**	**-68.70**

Method	Split CIFAR-100 (Task-Increment)							
Fine-tune	33.89±3.14							
Joint	91.63±0.06							
Buffer size	$\|\mathcal{M}\| = 500$				$\|\mathcal{M}\| = 1000$			
	A_1	A_∞	A_{m}	BWT	A_1	A_∞	A_{m}	BWT
GDUMB	-	36.40±0.97	-	-	-	43.25±0.35	-	-
GEM	85.53±2.30	68.68±0.99	68.49±1.58	-18.72	85.24±2.28	73.71±0.42	72.59±2.12	-12.81
AGEM	85.97±1.27	55.28±1.04	58.23±1.19	-34.10	85.66±1.84	55.95±1.99	59.96±1.58	-33.02
HAL	67.64±1.94	44.98±1.86	50.79±1.40	-25.17	68.62±2.93	50.07±2.34	54.01±2.47	-20.62
MIR	87.42±1.30	66.18±1.25	67.43±1.90	-23.60	87.50±1.23	71.20±0.60	71.42±1.46	-18.11
GSS	86.10±1.89	66.80±0.54	66.55±1.89	-21.45	86.44±1.85	71.98±0.72	71.00±1.81	-16.07
GMED	87.30±1.41	68.82±0.80	72.66±1.86	-20.53	87.49±1.64	73.91±0.35	76.36±1.82	-15.10
ER	87.29±1.65	66.82±1.04	67.56±1.68	-22.74	87.40±1.50	71.74±0.55	71.60±1.90	-17.40
Ours	**88.94±0.80**	70.03±0.57	74.07±0.94	-21.01	88.94±0.73	75.32±0.43	78.09±0.97	-15.14
Ours+RehSel	88.58±0.82	**70.81±0.76**	**74.24±0.90**	**-19.75**	**89.03±0.65**	**76.14±0.88**	**78.27±0.89**	**-14.33**

D Validation Sets Size

In Fig. 1, we show the validation size effect in MetaSP. For three metrics, when the validation size grows, the value gets larger, which means that a large validation set will improve SP.

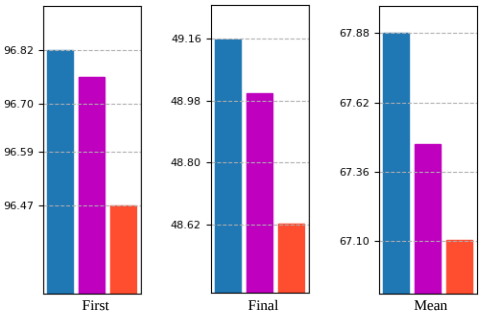

Figure 1: Performance with different validation sets size.

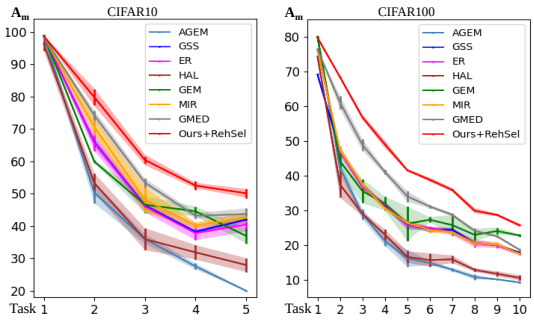

Figure 2: Learning processes on Split-CIFAR-10 and Split-CIFAR-100.

E Training Process

We also visualize the CL training process on Split CIFAR-10 and Split CIFAR-100 in Fig. 2. The first observation is that the proposed MetaSP outperforms other methods a lot, which means better SP throughout the CL training. Second, the forgetting cannot be eliminated even in the rehearsal-based CL. Even so, MetaSP offers an elegant add-in for the rehearsal-based CL methods and can further improve performance.

Table 4: Comparisons on Split Mini-Imagenet, averaged across 5 runs. Red and blue values mean the best in our methods and the previous methods. • indicates that our method is significantly better than the compared method (paired t-tests at 95% significance level).

Method	Split Mini-Imagenet (Class-Increment)							
Fine-tune	11.12±0.23							
Joint	44.39±0.74							
Buffer size	$\|\mathcal{M}\| = 500$				$\|\mathcal{M}\| = 1000$			
	A_1	A_∞	A_{m}	BWT	A_1	A_∞	A_{m}	BWT
GDUMB	-	6.22±0.27	-	-	-	7.15±1.96	-	-
AGEM	50.06±0.42	10.69±0.07	22.29±0.23	-49.22	50.03±0.31	10.69±0.22	22.28±0.09	-49.17
MIR	51.44±0.40	11.07±0.22	23.65±0.16	-50.46	51.25±0.37	11.32±0.18	24.09±0.18	-49.92
GSS	51.63±0.51	11.09±0.12	23.62±0.24	-50.67	51.35±.36	11.42±0.22	24.05±0.19	-49.91
GMED	51.21±0.69	11.03±0.18	24.47±0.29	-50.42	50.87±0.28	11.73±0.23	25.50±0.15	-49.93
ER	51.68±0.53	11.00±0.24	23.71±0.21	-50.84	51.41±0.60	11.35±0.35	24.08±0.31	-50.08
Ours	51.76±0.12	12.48±0.12	**26.50±0.15**	-49.10	50.91±0.41	14.43±0.31	**28.47±0.18**	-45.59
Ours+RehSel	**51.81±0.39**	**12.74±0.17**	26.43±0.11	**-48.85**	50.96±0.14	**14.54±0.22**	28.44±0.20	**-45.52**
Method	Split Mini-Imagenet (Task-Increment)							
Fine-tune	23.46±1.17							
Joint	62.3±0.48							
Buffer size	$\|\mathcal{M}\| = 500$				$\|\mathcal{M}\| = 1000$			
	A_1	A_∞	A_{m}	BWT	A_1	A_∞	A_{m}	BWT
GDUMB	-	16.37±0.37	-	-	-	17.69±3.24	-	-
AGEM	50.06±0.42	18.34±0.56	28.05±0.37	-39.66	50.03±0.31	18.78±0.54	28.12±0.30	-39.06
MIR	51.47±0.40	29.10±0.52	35.20±0.46	-27.96	51.31±0.37	31.39±0.44	37.24±0.52	-24.90
GSS	51.64±0.51	28.67±0.66	35.22±0.60	-28.72	51.40±0.35	31.75±0.71	37.23±0.48	-24.56
GMED	51.29±0.68	30.47±0.39	37.64±0.59	-26.03	51.00±0332	32.85±0.27	39.66±0.36	-22.69
ER	51.70±0.52	28.97±0.36	35.30±0.43	-28.41	51.55±0.57	31.59±0.78	37.36±0.57	-24.95
Ours	**52.44±0.20**	32.59±1.09	39.38±0.37	-24.82	**52.27±0.40**	36.25±0.27	41.59±0.33	-20.03
Ours+RehSel	51.73±0.41	**34.36±0.28**	**40.48±0.42**	**-21.71**	51.47±0.59	**37.20±0.73**	**42.19±0.51**	**-17.83**

F Time Analysis

Implemented epochs. In our implementation, we set 50 epochs in total for training each task, in which the first 45 epochs are naive fine-tuning and the last 5 epochs are with the proposed methods. As shown in Fig. 3(a), our method is implemented from the last 1 to 8 epochs. As the extra experiments use naive fine-tuning, where the new task is trained without interference. We prefer to evaluate if new tasks will be affected by memory. Easy to see, from last 1 to last 7, the performances of new task A1 get improved without a break, and from last 5, the performance barely grows anymore. This indicates that interference from old task to new task becomes hard from the last 1 to the last 5, and gets balance from the last 5 to 8. However, more epochs mean more computation costs. Thus, in our experiment, we leverage our methods in the last 5 epochs as a trade-off between efficiency and accuracy.

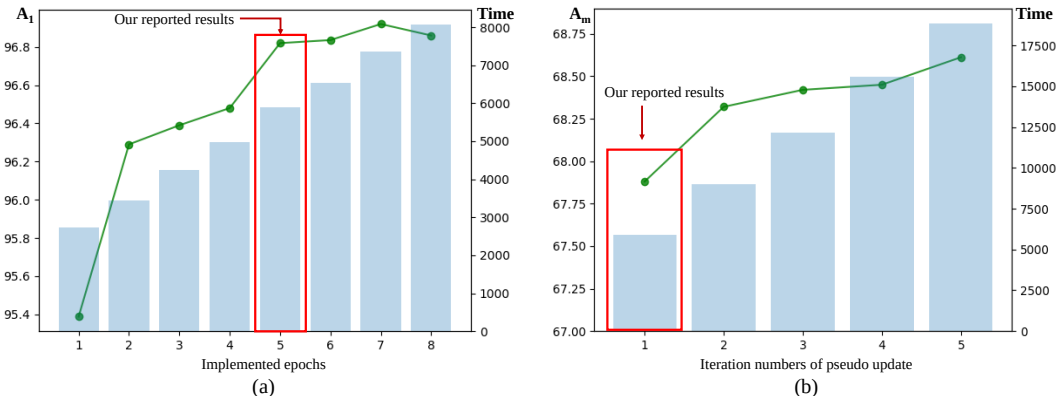

Figure 3: (a) Comparisons on different numbers of implemented epochs using only one-step pseudo update, from the last 1 to 8 epochs. (b) Comparisons on different numbers of pseudo update using 5 implemented epochs, from 1 to 5 iterations. The two subfigures are with the same legend, where the green lines represent the accuracy while blue bars mean the training time.

Table 5: Rehearsal comparisons.

Method	Split CIFAR-10						Split CIFAR-100					
	Class-Increment			Task-Increment			Class-Increment			Task-Increment		
	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m	A_1	A_∞	A_m
Random	96.82	49.16	67.88	97.30	90.91	93.38	88.13	18.96	38.62	88.94	70.03	74.07
RehSel (w/o cluster)	96.60	38.50	62.38	97.27	85.55	91.22	87.54	16.40	35.83	88.35	65.83	70.67
RehSel (w/o influence)	96.82	47.55	67.49	97.15	90.33	93.07	87.96	19.04	38.90	88.78	70.32	74.18
RehSel	96.81	50.10	68.28	97.30	91.41	93.29	87.81	19.28	39.23	88.58	70.81	74.24

Table 6: Influence approximation comparisons.

Method	Total examples	True Positive	True Negative	False Positive	False Negative
Larsen [3]	1000	552	304	26	118
Luketina [5]	1000	533	313	17	137
Neumann Series [4]	1000	642	282	48	28
Ours	1000	650	315	15	20

Iteration number for pseudo update. To compute example influence, we need to compute an argmin problem

$$\hat{\boldsymbol{\theta}}_{\mathbf{E},x} \coloneqq \arg\min_{\boldsymbol{\theta}} \ell\left(\mathcal{B}, \boldsymbol{\theta}\right) + \mathbf{E}^\top \mathbf{L}(x, \boldsymbol{\theta}), \quad x \in \mathcal{B}.$$

In Fig. 3(b), we show the results with different iterations to solve the problem. Experiments are implemented under the last 5 epochs. Easy to observe, a large number of iterations is equal to better performance but increases computational time linearly. Thus, we set only one iteration to keep the efficient training in compared with other CL methods' time.

G Comparisons with Other Influence-based Rehearsal Methods

In this paper, we proposed an influence-based rehearsal selection method under fixed budget memory $\mathcal{M}$. After task t finishes its training, for storing, we first cluster all training data into $\frac{|\mathcal{M}|}{t}$ groups using K-means to diversify the store data. Each group is ranked by their SP influence expection, *i.e.*, $\mathbb{E}(I^*(x))$, and the most positive influence on both SP will be selected to store. For dropping, we rank again on memory buffer via their influence values, and drop the most negative $\frac{|\mathcal{M}|}{t}$ example.

We also show if we do not cluster but select the top $\frac{|\mathcal{M}|}{t}$ examples only via their influence on SP (w/o clustering). The results are shown in Table 5. Using examples with only larger influence makes the selection concentrate on some saliency examples and lack the example diversity. Moreover, we evaluate if we only cluster and select the nearest examples instead of the example influence (w/o influence). Finally, with both influence ranking and cluster, the selected examples make the training forget less.

H Details of Influence Statistics in Fig. 3 of This Paper

As shown in Fig. 3 in the paper, we count the example with positive and negative influence on old task (S), new task (P) and total SP and show the distribution in Split-CIFAR-10. Here, we give the details of these experiments. The experiments are with 50 implemented epochs and 5 pseudo-update iterations. We do not use the update and rehearsal selection strategy. For each task from task 2, we have 500 fixed-size memory and 10,000 new task data. We divide all example influences equally into 5 groups from the minimum to the maximum.

I MGDA, KKT Conditions and The Solution

In Eq. (10) of this paper, we introduce the dual objective problem and the Multiple-Gradient Descent Algorithm (MGDA) [1]. To obtain the objective, we first introduce the Steepest Gradient Method (SGM) [2] in dual-objective optimization. Given two tasks 1 and 2, the objective of SGM is

$$\mathbf{d}^*, \alpha^* = \arg\min_{\mathbf{d},\alpha} \quad \alpha + \frac{1}{2}||\mathbf{d}||^2, \quad \text{s.t.} \quad \mathbf{g}_1^\top \mathbf{d} \leq \alpha \text{ and } \mathbf{g}_2^\top \mathbf{d} \leq \alpha,$$

where $\mathbf{g}_1$ and $\mathbf{g}_2$ are the gradients for tasks 1 and 2 specifically. The two constraints can be seen as the difference between task gradients and the optimal gradients.

Considering the Lagrange multipliers λ_1 and λ_2 for the two constraints, we have the dual problem of the above problem as

$$\lambda_1^*, \lambda_2^* = -\max_{\lambda_1,\lambda_2} ||\lambda_1 \mathbf{g}_1 + \lambda_2 \mathbf{g}_2||^2, \quad \text{s.t.} \quad \lambda_1 + \lambda_2 = 1 \text{ and } \lambda_1 \geq 0, \lambda_2 \geq 0.$$

This is the objective of Eq. (10) of this paper, *i.e.*, MGDA [1]. In SGM, the KKT conditions can be written as

$$\begin{aligned} \lambda_1^*(\mathbf{g}_1^\top \mathbf{d}^* - \alpha^*) &= 0, \\ \lambda_2^*(\mathbf{g}_2^\top \mathbf{d}^* - \alpha^*) &= 0, \\ \lambda_1^* \geq 0, \lambda_2^* &\geq 0, \\ \lambda_1^* + \lambda_2^* &= 1, \\ \lambda_1^* \mathbf{g}_1 + \lambda_2^* \mathbf{g}_2 &= \mathbf{d}^*. \end{aligned}$$

The solution to the dual problem is

$$\gamma^* = \min\left(\max\left(\frac{(\mathbf{g}_2 - \mathbf{g}_1)^\top \mathbf{g}_2}{\|\mathbf{g}_2 - \mathbf{g}_1\|_2^2}, 0\right), 1\right).$$

This is the solution in Eq. (11) of this paper.

J Comparisons on Influence Function Approximation

To evaluate the example influence approximation from our meta method to Hessian influence function, we build a toy experiment compared with three extra baselines. The three baselines [3, 5, 4] use different ways to approximate inverse Hessian. We use 1000 FMNIST training data and 500 test data. We design a simple fc network with a single hidden layer. The results are the influence from the 1000 training data to 500 test data. We have the following observations: (1) Most examples (965/1000) have the true influence property compared with Hessian influence function; (2) The results show the proposed method has a better approximation rate compared with other inverse Hessian approximation methods.

References

[1] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 2012.

[2] Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 2000.

[3] Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*. IEEE, 1996.

[4] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020.

[5] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *ICML*, 2016.