Learning with Recoverable Forgetting —*Supplemental Material*—

Jingwen Ye[1], Yifang Fu[1], Jie Song[2], Xingyi Yang[1], Songhua Liu[1], Xin Jin[3], Mingli Song[2], and Xinchao Wang[1†]

[1] National University of Singapore
[2] Zhejiang University [3] Eastern Institute of Advanced Study
{jingweny,xinchao}@nus.edu.sg, {e0724403,xyang,songhua.liu}@u.nus.edu
{sjie,brooksong}@zju.edu.cn, jinxin@eias.ac.cn

In this document, we provide the supplementary materials that cannot fit into the main manuscript due to the page limit. Specifically, we give more details on the proposed LIRF model, alongside more experimental results.

1 More Details of LIRF

1.1 Network Pruning

To apply 'selective knowledge damage' on deposit module $\mathcal{T}_r$, we use the ranking pruning method to get the initialized light-wight deposit module from the original network $\mathcal{T}_0$:$\mathcal{T}_r \xleftarrow{Initialize} \mathcal{P}rune[\mathcal{T}_0^{(-n)}]$. The proposed network pruning here can also be treated as a partial-knowledge-transfer technique to transfer the sample-specific knowledge of the deposit $\mathcal{D}_r$.

To be more specific, we use a ranking-based pruning method [3] that prunes the filters with lowest $\ell 1$-norm values. Let n_i be the input channel and n_o be the output channel , then the filters at that layer can be denoted as n_i 2D kernels $\mathcal{K} \in \mathbb{R}^{k\times k}$ (e.g. 3×3). We measure the relative importance of a filter in each layer by calculating the sum of its absolute weights:

$$\mathcal{S}_j(\mathcal{K}) = \sum_{l}^{n_i} |\mathcal{K}_l|. \tag{1}$$

Then we sort the filters by $\mathcal{S}_j$, which is used to prune the filters with the smallest sum values and their corresponding feature maps. Also, the kernels in the next convolutional layer corresponding to the pruned feature maps are removed.

Since this value also represents the average magnitude of its kernel weights, it gives an expectation of the magnitude of the output feature map. Filters with smaller kernel weights tend to produce feature maps with weak activations, as compared to the other filters in that layer. Thus we retain the high-activated filters, which are expected to contain the sample-specific knowledge.

† Corresponding author.

1.2 Algorithm

The whole algorithm of LIRF is given in Alg. 1, which includes the knowledge deposit and knowledge withdrawal processes.

Algorithm 1 Learning with Recoverable Forgetting

———*Knowledge Deposit*———

Require: $\mathcal{D}_r$: deposit set; $\mathcal{T}_0$: original network trained on the full dataset; n: the block to divide the networks.

Ensure: $\mathcal{T}$: the target network; $\mathcal{T}_r$: the deposit module.

Divide the original network into two modules: $\mathcal{T}_0 = \mathcal{T}_0^{(-n)} \circ \mathcal{T}_0^{(n-)}$;

Separate and initialize the target network as $\mathcal{T} = \mathcal{T}^{(-n)} \circ \mathcal{T}^{(n-)}$, where $\mathcal{T}^{(-n)} \leftarrow \mathcal{T}_0^{(-n)}$ and $\mathcal{T}^{(n-)} \leftarrow \mathcal{T}_0^{(n-)}$;

Initialize the deposit module with pruning: $\mathcal{T}_r^{(-n)} \leftarrow \mathcal{P}rune[\mathcal{T}_0^{(-n)}]$;

repeat

  Input $x \subset \mathcal{D}_r$ to $\mathcal{T}_0$, $\mathcal{T}$ and $\mathcal{T}_r$;

  Generate random labels y_r for the input x;

  Calculate the knowledge removal loss $\mathcal{L}_{kr}$;

  Calculate the knowledge preservation loss $\mathcal{L}_{kp}$;

  Calculate the partial knowledge transfer loss $\mathcal{L}_{pt}$;

  Calculate the recover loss $\mathcal{L}_{re}$;

  Update the parameters of $\mathcal{T}^{(-n)}$ and $\mathcal{T}_r$ by the total loss $\mathcal{L}_{all}$;

until Convergence

Return the target network $\mathcal{T}$ and store the deposit module $\mathcal{T}_r$.

———*Knowledge Withdrawal*———

Require: $\mathcal{T}$: target network; $\mathcal{T}_r$: deposit module;

Ensure: $\widetilde{\mathcal{T}}$: the recover network.

For input x, construct the recover network by: $\widetilde{\mathcal{T}}(x) = g\big(\mathcal{T}(x)\big) + \overline{g}\big(\mathcal{T}_r \circ \mathcal{T}^{(n-)}(x)\big)$;

Return the recover network $\widetilde{\mathcal{T}}$.

1.3 LIRF with Multiple Deposit Modules

In the main paper, we discuss the case on depositing and withdrawing one deposit set at one time. In fact, it can be readily extended to depositing multiple deposit sets $\{\mathcal{D}_r^1, \mathcal{D}_r^2, ..., \mathcal{D}_r^m\}$ to multiple deposit modules $\{\mathcal{T}_r^1, \mathcal{T}_r^2, ..., \mathcal{T}_r^m\}$. The process could be formulated as:

$$\mathcal{T}_0 \xrightarrow[\{\mathcal{D}_r^1, \mathcal{D}_r^2, ..., \mathcal{D}_r^m\}]{\text{Deposit}} \{\mathcal{T}, \{\mathcal{T}_r^1, \mathcal{T}_r^2, ..., \mathcal{T}_r^m\}\} \xrightarrow{\text{Withdraw}} \widetilde{\mathcal{T}}. \tag{2}$$

Thus, there are two ways to deposit the sample-related knowledge to multiple modules, which are 'depositing once' and 'depositing in sequence'.

Depositing once. Depositing the multi-source knowledge once means that we transfer the sample-specific knowledge to each deposit module $\{\mathcal{T}_r^1, \mathcal{T}_r^2, ..., \mathcal{T}_r^m\}$ with training the LIRF framework once.

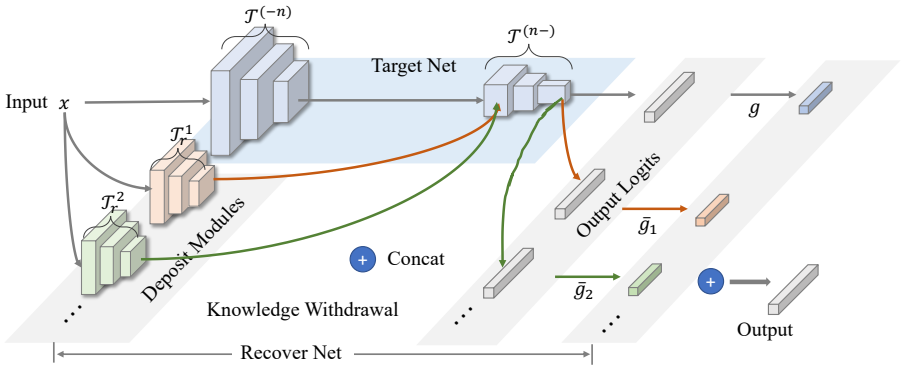

Fig. 1. The form of recover net in the knowledge withdrawal process. The deposit modules and the target net are together to build the recover net.

Recall that the total loss is $\mathcal{L}_{all} = \mathcal{L}_{kr} + \lambda_{kp}\mathcal{L}_{kp} + \lambda_{re}\mathcal{L}_{re} + \lambda_{pt}\mathcal{L}_{pt}$, then the loss function can be rewritten with multi-deposit sets as:

$$\begin{aligned}
\mathcal{L}_{kr} &= \mathcal{L}_{ce}\big(\mathcal{T}(x), y_r\big) - \lambda_{at}\mathcal{L}_{at}\big(\mathcal{T}^{(-n)}(x), \mathcal{T}_0^{(-n)}(x)\big), \\
\mathcal{L}_{kp} &= \mathcal{L}_{kd}\big(g(\frac{z_{\mathcal{T}}(x)}{T}), g(\frac{z_{\mathcal{T}_0}(x)}{T})\big), \\
\mathcal{L}_{re} &= \sum_i \mathcal{L}_{ce}\big(\widetilde{\mathcal{T}}(x_i), y_i\big), \\
\mathcal{L}_{pt} &= \sum_i \mathcal{L}_{kd}\big(\bar{g}_i(\frac{z_{\mathcal{T}_r^i \circ \mathcal{T}^{(n-)}}(x_i)}{T}), \bar{g}_i(\frac{z_{\mathcal{T}_0}(x_i)}{T})\big), \\
&\text{w.r.t} \quad x \subset \bigcup_i \mathcal{D}_r^i \quad \text{and} \quad x_i \subset \mathcal{D}_r^i,
\end{aligned} \tag{3}$$

where g_i is the filter to select D_r^i-related logits and g is the filter to select the logits related to the preservation set $\bigcap_i \overline{\mathcal{D}_r^i}$.

Depositing in sequence. Depositing the knowledge in sequence means that the we deposit one sample-specific knowledge that related to $\mathcal{D}_r^i (1 \le i \le m)$ at one time. For example, when we transfer the knowledge from $\mathcal{T}_0$ to $\mathcal{T}_r^1$ with loss $\mathcal{L}_{all}$ dealing with one deposit set $\mathcal{D}_r^1$, at the next step to deposit $\mathcal{D}_r^2$, the former target net is treated as the new original network:

$$\mathcal{T}_0 \xrightarrow[\mathcal{D}_r^1]{\text{Deposit}} \{\mathcal{T}^1, \mathcal{T}_r^1\}; \mathcal{T}^1 \xrightarrow[\mathcal{D}_r^2]{\text{Deposit}} \{\mathcal{T}^2, \mathcal{T}_r^2\}; ...; \mathcal{T}^{m-1} \xrightarrow[\mathcal{D}_r^m]{\text{Deposit}} \{\mathcal{T}, \mathcal{T}_r^m\}, \tag{4}$$

which means that the LIRF is trained m times to deposit all sets. This proves that the proposed method can deal with the incoming requires for depositing.

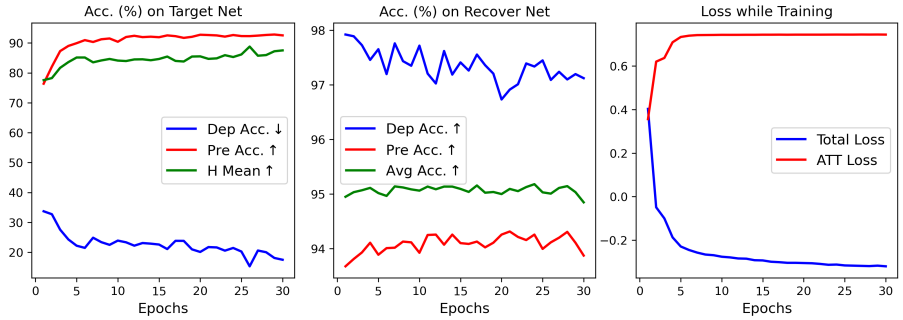

Fig. 2. The accuracy curves ('Pre Acc.', 'Dep Acc.', 'H Mean' and 'Avg Acc.') and loss curves ('Total Loss' and 'ATT Loss') while training in LIRF. (Note a total of 20 epochs are used for training.)

Knowledge withdrawal from multiple deposit modules. Given the deposit module set $\{\mathcal{T}_r^1, \mathcal{T}_r^2, ..., \mathcal{T}_r^m\}$ and the target network $\mathcal{T}$, the recover network is reorganized as:

$$\widetilde{\mathcal{T}}(x) = g\big(\mathcal{T}(x)\big) + \sum_i \overline{g}_i\big(\mathcal{T}_r^i \circ \mathcal{T}^{(n-)}(x)\big), \tag{5}$$

where g is the filter to select the logits related to the preservation set. The process of knowledge withdrawal to the recover net is depicted in Fig. 1.

2 More Experiments

The loss curve and accuracy curve while training. In Fig. 2, we depict the curves of 'Dep Acc.', 'Pre Acc.' 'H Mean' on the target net and 'Dep Acc.', 'Pre Acc.' and 'Avg Acc.' on the recover net, as well as the total loss ($\mathcal{L}_{all}$) and att loss ('$\mathcal{L}'_{at}$) while training the whole LIRF framework. As can be seen in the figure, the losses convergence first in the first few epochs, where the we maximise the 'ATT loss' ($\mathcal{L}_{at}$) in the training process. Thus, the proposed FIRF is time-efficient, 20 epochs of training is enough for finetuning the whole framework. Also, the accuracy on the preservation set drops slightly due to bias caused by training only with the deposit set $\mathcal{D}_r$. However, this drop is not sufficiently significant, which doesn't affect the average accuracy ('Avg Acc.') on the whole dataset.

2.1 More Ablation Study on LIRF

The influence of the deposit set scale. In the normal setting of the proposed framework, 20% of the data is selected to form the deposit set $\mathcal{D}_r$. Here, to explore the influence of the scale of the deposit set, we deposit different percentages of the deposit set and show the performances on the target network and the recover network, as shown in Fig. 1. We observe that:

Table 1. Experimental results of the ablation study on the scale of the deposit set. The experiments are conducted on CIFAR-10 dataset.

Scale ($\lvert\mathcal{D}_r\rvert/\lvert\mathcal{D}\rvert$)	#Target Net			#Recover Net		
	Pre Acc. ↑	Dep Acc. ↓	H Mean↑	Pre Acc. ↑	Dep Acc. ↑	Avg Acc. ↑
Deposit-10%	93.18	27.93	77.27	92.73	99.81	93.44
Deposit-20%	93.97	16.64	86.30	93.61	98.13	94.51
Deposit-30%	93.42	14.15	86.45	94.55	97.67	95.49
Deposit-40%	95.91	21.29	86.21	95.47	95.63	95.53
Deposit-50%	97.13	19.20	84.22	95.86	95.27	95.57
Deposit-60%	98.12	11.80	88.74	96.61	94.15	95.13
Deposit-70%	98.31	9.17	90.64	97.08	94.02	94.94
Deposit-80%	96.41	13.67	87.40	96.91	94.16	94.71
Deposit-90%	64.52	2.75	75.56	95.76	94.08	94.25

– No matter how large the deposit set is, the proposed LIRF works in both knowledge deposit and knowledge withdrawal (high 'H Mean' and 'Avg Acc.');
– With the increase of the deposit scale ($|\mathcal{D}_r|/|\mathcal{D}|$), 'Pre Acc.' on the target net increases firstly because of more training data to boot the partial knowledge transfer, and drops at last due to the bias caused by the deposit sets.
– Depositing nearly half of the full data has the best performance on both the knowledge deposit and withdrawal processes (the highest high 'H Mean' and 'Avg Acc.')
– When deposit 90% data, 'Pre Acc.' on the target net turns to be undesirable. This is however understandable since, when there is a request to deposit almost all of the data, the model performance downgrades and it is better to retrain a new model on the preserved dataset.

2.2 LIRF with multiple deposit modules.

As discussed in Sec. 1.3, we explore two means of depositing the knowledge with multiple deposit modules, which are depositing once and depositing in sequence.

Different deposit orders. Here, we compare the performance of these two multi-deposit methods. Let $\mathcal{D}_r^1$ and $\mathcal{D}_r^2$ be two deposit sets (each contains 20% of the full data $\mathcal{D}$), then the comparative results are displayed in Table 2. The deposit orders for comparison are:

$$\begin{aligned}
[\text{Once}] &: \mathcal{T}_0 \xrightarrow[\{\mathcal{D}_r^1,\mathcal{D}_r^2\}]{\text{Deposit}} \{\mathcal{T}, \{\mathcal{T}_r^1, \mathcal{T}_r^2\}\}, \\
[\text{Sequence}] &: \mathcal{T}_0 \xrightarrow[\mathcal{D}_r^1]{\text{Deposit(1)}} \{\mathcal{T}_1, \mathcal{T}_r^1\} \xrightarrow[\mathcal{D}_r^2]{\text{Deposit(2)}} \{\mathcal{T}, \mathcal{T}_r^2\}, \\
[\text{Sequence-inv}] &: \mathcal{T}_0 \xrightarrow[\mathcal{D}_r^2]{\text{Deposit(1)}} \{\mathcal{T}_1, \mathcal{T}_r^2\} \xrightarrow[\mathcal{D}_r^1]{\text{Deposit(2)}} \{\mathcal{T}, \mathcal{T}_r^1\}.
\end{aligned} \tag{6}$$

Table 2. Experimental results on the LIRF with multiple deposit modules. For the process of knowledge deposit we compute 'H Mean' and for knowledge withdrawal, we compute 'Avg Acc.'.

Order	Process	Pre Acc. ↑	$\mathcal{D}_r^1$ Acc.	$\mathcal{D}_r^2$ Acc.	H Mean/Avg Acc.↑
Original	-	94.80	96.43	90.58	94.28
Once	Deposit	95.41	15.37	17.64	85.22
	Withdraw	94.94	96.20	97.06	95.62
Sequence	Deposit(1)	93.97	16.64	-	86.30
	Deposit(2)	93.19	-	14.48	83.78
	Withdraw	93.01	97.23	96.56	94.36
Sequence-inv	Deposit(1)	93.85	-	19.17	81.11
	Deposit(2)	93.92	14.37	-	87.62
	Withdraw	93.26	97.96	95.83	94.71

From the figure, several conclusions could be drawn that:

- All the three orders ('Once', 'Sequence' and 'Sequence-inv') are capable of depositing and withdrawing the sample-specific knowledge of each deposit set;
- In the proposed multi-deposit LIFR framework, the scheme 'depositing once' shows better performance than the scheme 'depositing in sequence'. It is mainly because that more data is included to finetune LIRF, which reduces that data bias;
- When depositing in sequence, the order of the deposit set doesn't affect the final performance.

Different withdrawal orders. Also, for comparing the recover net with different orders to withdraw from the deposit module set, we conduct the experiments depicted in Tabel 4. The withdrawal orders for comparison are:

$$\begin{aligned}
\text{[Once]} &: \{\mathcal{T}, \{\mathcal{T}_r^1, \mathcal{T}_r^2\}\} \xrightarrow[\{\mathcal{D}_r^1, \mathcal{D}_r^2\}]{\text{Withdraw}} \widetilde{\mathcal{T}}, \\
\text{[Sequence]} &: \{\mathcal{T}, \mathcal{T}_r^1\} \xrightarrow{\text{Withdraw(1)}} \widetilde{\mathcal{T}}_1; \quad \{\widetilde{\mathcal{T}}_1, \mathcal{T}_r^2\} \xrightarrow{\text{Withdraw(2)}} \widetilde{\mathcal{T}}, \\
\text{[Sequence-inv]} &: \{\mathcal{T}, \mathcal{T}_r^2\} \xrightarrow{\text{Withdraw(1)}} \widetilde{\mathcal{T}}_1; \quad \{\widetilde{\mathcal{T}}_1, \mathcal{T}_r^1\} \xrightarrow{\text{Withdraw(2)}} \widetilde{\mathcal{T}}.
\end{aligned} \tag{7}$$

From the table, we observe that:

- All the deposit orders don't affect the performance of the final recover net, since the final formulation of the recover net wouldn't be changed;
- The deposit orders don't affect the performance of the intermediate recover net ($\widetilde{\mathcal{T}}_1$) a lot. So in the proposed LIRF, it is flexible to withdraw from a set of deposit modules.

Table 3. Experimental results on the LIRF with multiple deposit modules. We compare the performances of the recover net with different withdrawal orders.

Method	Process	Pre Acc.↑	$\mathcal{D}_r^1$ Acc.↑	$\mathcal{D}_r^2$ Acc.↑	Avg Acc.↑
Original	-	94.80	96.43	90.58	94.28
Once	Withdraw	94.94	96.20	97.06	95.62
Sequence	Withdraw(1)	95.22	96.89	-	95.64
	Withdraw(2)	94.94	96.20	97.06	95.62
Sequence-inv	Withdraw(1)	95.29	-	97.46	95.83
	Withdraw(2)	94.94	96.20	97.06	95.62

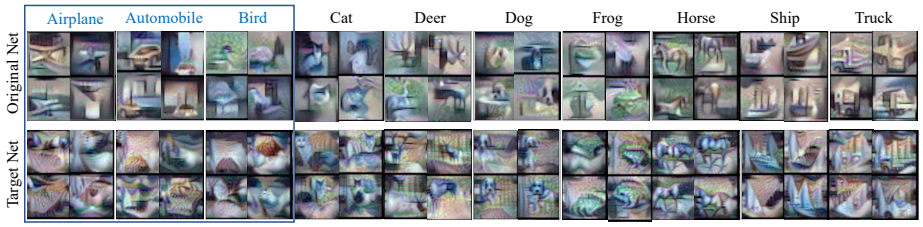

Fig. 3. Images generated by inverting the original network ($\mathcal{T}_0$) and the target network ($\mathcal{T}$) trained on CIFAR-10 with DeepInversion. For each image, each group containing 2×2 images represents one category. The deposit set is marked in blue.

2.3 Privacy Test via Data-free setting

To verify whether the proposed LIRF has removed the sample-specific knowledge from the target network, we evaluate the target net $\mathcal{T}$ on the data-free knowledge distillation method DeepInversion [8], where we treat the target net as teacher and students only have access to the probabilities produced by teacher networks.

DeepInversion inverts a trained network (teacher) to synthesize class-conditional input images starting from random noise, without using any additional information about the training dataset. By DeepInversion, we transfer the knowledge from the teacher network to the image domain, which could be thought as a kind of visualization of the sample-specific knowledge. The synthetic images from the original net ($\mathcal{T}_0$) and the target net ($\mathcal{T}$) are compared in Fig. 3, where the first three categories are in the deposit set. As can be seen from the figure, the images synthesized from the original net show the clear main object of the category, which show that the sample-specific knowledge of all categories is well-preserved in the original network. It is similar in the preservation categories of the target net. While in target net, the images synthesized from the deposit categories are hard to be recognized by human eyes, which shows that we have already removed the sample-specific knowledge from the target net.

Table 4. Comparative experimental results on incremental learning.

Method	$s = 5$	$s = 10$	$s = 20$	$s = 50$
LwF [4]	29.5	40.4	47.6	52.9
iCaRL [5]	57.8	60.5	62.0	61.8
EEIL [1]	63.4	63.6	63.7	60.8
BiC [7]	60.1	60.4	68.9	70.2
LIRF(ours)	72.6	74.9	77.0	79.4

Table 5. Comparative experimental results on machine unlearning.

Method	#1-Class			#2-Class		
	Pre Acc. ↑	Dep Acc. ↓	H Mean↑	Pre Acc. ↑	Dep Acc. ↓	H Mean↑
Original	84.05	87.49	0	84.18	84.72	0
Retrain	85.72	0	86.60	86.30	0	85.50
Min-Max [2]	20.48	5.11	32.64	29.59	7.59	42.77
GKT [2]	81.97	0	84.64	81.70	0	83.18
LIRF(Our)	84.19	12.30	79.43	84.07	15.82	75.73

2.4 Comparing with Incremental Learning

The proposed LIRF could be modified to do the class-incremental task. Here, we compare the proposed framework with the other incremental learning methods. The experiments are conducted on CIFAR-100 dataset. Please also note that the setting of the proposed LIRF is different from the normal incremental learning methods. In order to adapt LIRF with the incremental setting, we train the framework in the following steps: (1) We train the original network $\mathcal{T}$ with all the C categories; (2) For the incremental step that contains s classes, we finetune LIRP with depositing in sequence, so that we have one target net and $(C/s-1)$ deposit modules; (3) For each incremental step, we withdraw from one deposit module. The comparative results are conducted on CIFAR-100 dataset, and are depicted in Table 4, where the methods for comparison are the standard incremental learning methods.

Comparing with the methods in Table 4, the proposed LIRF outperforms others by a large margin. It is because that the LIRF trains the original network firstly and then does the knowledge deposit. Here, please note that we do not want to claim that we propose the state-of-the-art method in incremental learning. The experiments are conducted here to prove that the good performance of knowledge withdrawal from the deposit modules. And when the data in deposit set is available again, the proposed LIRF performs much better than the methods where the network is trained with treating this data as the new-coming data. Thus, it again shows the necessarily to deposit the knowledge, instead of directly abandoning it.

2.5 Comparing with Machine Unlearning

In order to compare with the state-of-the-art machine unlearning methods, we follow the setting of the previous unlearning method [2]. We demonstrate the performance of the proposed methods for unlearning single and multiple classes on CIFAR-10 dataset and compare the proposed LIRF with the methods that work on the class-level unlearning. For fair comparison, we unlearn class 0 in 1-class and classes 1 and 2 in 2-classes unlearning and use ALLCNN [6] as backbone. The comparative results are displayed in Table 5. The proposed LIRF also performs good in the field of machine unlearning, which demonstrate the generalization ability of LIRF. Also, it can be observed from the table that LIRF does well in preserving the performance of the preservation set.

References

1. Castro, F.M., Marín-Jiménez, M.J., Mata, N.G., Schmid, C., Alahari, K.: End-to-end incremental learning. In: European Conference on Computer Vision (2018)
2. Chundawat, V.S., Tarun, A.K., Mandal, M., Kankanhalli, M.S.: Zero-shot machine unlearning. ArXiv **abs/2201.05629** (2022)
3. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets (2016)
4. Li, Z., Hoiem, D.: Learning without forgetting. In: European Conference on Computer Vision (2016)
5. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. IEEE Conference on Computer Vision and Pattern Recognition pp. 5533–5542 (2017)
6. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. CoRR **abs/1412.6806** (2015)
7. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.R.: Large scale incremental learning. Conference on Computer Vision and Pattern Recognition pp. 374–382 (2019)
8. Yin, H., Molchanov, P., Li, Z., Álvarez, J.M., Mallya, A., Hoiem, D., Jha, N.K., Kautz, J.: Dreaming to distill: Data-free knowledge transfer via deepinversion. Conference on Computer Vision and Pattern Recognition pp. 8712–8721 (2020)