# Mnemonics Training: Multi-Class Incremental Learning without Forgetting
## *Supplementary Materials*

Yaoyao Liu[1,2*] Yuting Su[1†] An-An Liu[1†] Bernt Schiele[2] Qianru Sun[3]

[1]School of Electrical and Information Engineering, Tianjin University
[2]Max Planck Institute for Informatics, Saarland Informatics Campus
[3]School of Information Systems, Singapore Management University
`{yaoyao.liu, schiele, qsun}@mpi-inf.mpg.de`
`{liuyaoyao, ytsu, liuanan}@tju.edu.cn qianrusun@smu.edu.sg`

These supplementary materials include the entire algorithm (§A), the details of weight transfer operations (§B), the experiments in uniform memory budget setting (§C), the ablation study for distillation loss (§D), comparison with generative replay method (§E), and more visualization results (§F). In addition, our open-source code is on GitHub[1].

## A. Algorithm

This is supplementary to Section 4. In Algorithm 1, we summarize the overall process of the proposed *mnemonics* training. Step 1-16 show the alternative learning of classification models and *mnemonics* exemplars, corresponding to Sections 4.1-4.3. Specifically in each phase, Step 8 executes the *model-level* training, while Step 11 and 14 are the *exemplar-level*. Step 17 is optional due to different MCIL settings regarding the memory budget. We conduct experiments in two settings: (1) each class has a fixed number (e.g., 20) of exemplars, and (2) the system consistently keeps a fixed memory budget in all phases, therefore, the system in earlier phases can store more exemplars per class and needs to discard old exemplars in later phases gradually. Step 18 fine-tunes the model on adjusted and balanced examples. It is helpful to reduce the previous model bias (Step 8) caused by the imbalance samples numbers between new class data $D_i$ and old exemplars $\mathcal{E}_{0:i-1}$. Step 19 is to evaluate the learned model $\Theta_i$ in the current phase, and the average over all phases will be reported as the final evaluation. Step 20 updates the memory to include new exemplars.

**Algorithm 1:** Mnemonics Training

```
Input: Data flow $\{D_i\}_{i=0}^{N}$.
Output: MCIL models $\{\Theta_i\}_{i=0}^{N}$, and mnemonics
        exemplars $\{\mathcal{E}_i\}_{i=0}^{N}$.
1  for $i$ in $\{0, 1, ..., N\}$ do
2      Get $D_i$;
3      if $i = 0$ then
4          Randomly initialize $\Theta_0$ and train it on $D_0$;
5      else
6          Get $\mathcal{E}_{0:i-1}$ from memory;
7          Initialize $\Theta_i$ with $\Theta_{i-1}$;
8          Train $\Theta_i$ on $\mathcal{E}_{0:i-1} \cup D_i$ by Eq. 6;
9      end
10     Sample $S$ from $D_i$ to initialize $\mathcal{E}_i$;
11     Train $\mathcal{E}_i$ using $\Theta_i$ by Eq. 9;
12     while $i \geq 1$ do
13         Split $\mathcal{E}_{0:i-1}$ into subsets $\mathcal{E}^{\mathbf{A}}_{0:i-1}$ and $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ ;
14         Optimize $\mathcal{E}^{\mathbf{A}}_{0:i-1}$ and $\mathcal{E}^{\mathbf{B}}_{0:i-1}$ by Eq. 10;
15         Get the adjusted old exemplars $\tilde{\mathcal{E}}_{0:i-1}$
16     end
17     (Optional) delete part of the exemplars in $\tilde{\mathcal{E}}_{0:i-1}$;
18     Finetune $\Theta_i$ on $\mathcal{E}_i \cup \tilde{\mathcal{E}}_{0:i-1}$;
19     Run test and record the results;
20     Update $\mathcal{E}_{0:i} \leftarrow \mathcal{E}_i \cup \tilde{\mathcal{E}}_{0:i-1}$ in memory.
21 end
```

## B. Weight transfer operations

This is supplementary to Section 5.1 “**the architecture of $\Theta$**”. We deploy weight transfer operations [4, 7] to train the weight scaling and shifting parameters (named $\mathbf{T}_i$) in the $i$-th phase which specifically transfer the network weights $\Theta_{i-1}$ to $\Theta_i$. The aim is to preserve the structural knowledge

---

[*] This work was done during Yaoyao's internship supervised by Qianru.
[†] Corresponding authors.
[1] Code: https://github.com/yaoyao-liu/mnemonics

of $\Theta_{i-1}$ when learning $\Theta_i$ on new class data.

Specifically, we assume the $q$-th layer of $\Theta_{i-1}$ contains $R$ neurons, so we have $R$ neuron weights and biases as $\{W_{q,r}, b_{q,r}\}_{r=1}^{R}$. For conciseness, we denote them as $W_q$ and $b_q$. For $W_q$, we learn $R$ scaling parameters denoted as $\mathcal{T}_q^W$, and for $b_q$, we learn $R$ shifting parameters denoted as $\mathcal{T}_q^b$. Let $X_{q-1}$ and $X_q$ be the input and output (feature maps) of the $q$-th layer. We apply $\mathcal{T}_q^W$ and $\mathcal{T}_q^b$ to $W_q$ and $b_q$ as,

$$X_q = (W_q \odot \mathcal{T}_q^W) X_{q-1} + (b_q + \mathcal{T}_q^b), \qquad \text{(S1)}$$

where $\odot$ donates the element-wise multiplication. Assuming there are $Q$ layers in total, the scaling and shifting parameters are denoted as $\mathbf{T}_i = \{\mathcal{T}_q^W, \mathcal{T}_q^b\}_{q=1}^{Q}$.

Therefore, to learn the MCIL model $\Theta_i$, we use the indirect way of training $\mathbf{T}_i$ (instead of the direct way of training $\Theta_i$) on $D_i \cup \mathcal{E}_{0:i-1}$ and keeping $\Theta_{i-1}$ fixed. During the training, both classification loss and distillation loss [2, 5] are used. Let $\odot_L$ donate the function of applying $\mathbf{T}_i$ to $\Theta_{i-1}$ by layers (Eq. S1). The objective function Eq. 4 in the main paper can be rewritten as:

$$\begin{aligned} \mathcal{L}_{\text{all}} =& \lambda \mathcal{L}_c(\mathbf{T}_i \odot_L \Theta_{i-1}; \mathcal{E}_{0:i-1} \cup D_i) \\ &+ (1-\lambda)\mathcal{L}_d(\mathbf{T}_i \odot_L \Theta_{i-1}; \Theta_{i-1}; \mathcal{E}_{0:i-1} \cup D_i), \end{aligned} \qquad \text{(S2)}$$

where $\lambda$ is a scalar manually set to balance two loss terms. Let $\alpha_1$ be the learning rate, $\mathbf{T}_i$ is updated with gradient descent as follows,

$$\mathbf{T}_i \leftarrow \mathbf{T}_i - \alpha_1 \nabla_{\mathbf{T}} \mathcal{L}_{\text{all}}. \qquad \text{(S3)}$$

After the learning of $\mathbf{T}_i$, we compute $\Theta_i$ as follows:

$$\Theta_i \leftarrow \mathbf{T}_i \odot_L \Theta_{i-1}. \qquad \text{(S4)}$$

## C. Uniform memory budget experiments

This is supplementary to Section 5.1 "**the architecture of $\mathcal{E}$**". As mentioned on Line 632-643 (in the main paper), we have two methods of setting memory budgets [1]. (1) **Uniform exemplar number setting (used in the main paper)**: every class has 20 exemplars. (2) **Uniform memory budget setting**: the system works on a uniform memory budget, e.g. $2,000$ $(20,000)$ total exemplars can be stored in each phase. In this setting, the system stores more exemplars per class in earlier phases and discards some of the old exemplars afterwards.

We have three strategies of discarding old exemplars: (1) ***random***: discarding old exemplars randomly to satisfy the limit of the budget; (2) ***high loss***: discarding old exemplars on which $\Theta_i$ has higher losses (i.e., harder exemplars); (3) ***low loss***: discarding old exemplars on which $\Theta_i$ has lower losses (i.e. easier exemplars).

| Exemplar | | *CIFAR-100* | | | *ImagNet-Subset* | | |
|---|---|---|---|---|---|---|---|
| | | *N*=5 | 10 | 25 | 5 | 10 | 25 |
| *random* | | 63.83 | 63.12 | 61.93 | 71.67 | 71.01 | 69.38 |
| *herding* | | 63.50 | 62.55 | 61.32 | 71.45 | 70.02 | 68.76 |
| ours *w/o* adj. | ↑ | 64.58 | 62.75 | 63.78 | 72.27 | 71.05 | 70.89 |
| ours | | **65.01** | **63.80** | **64.42** | **72.82** | **72.20** | **72.31** |
| *random* | | 18.50 | 15.64 | 16.09 | 16.87 | 17.12 | 17.41 |
| *herding* | | 20.43 | 18.44 | 19.60 | 17.34 | 17.95 | 17.59 |
| ours *w/o* adj. | ↓ | 12.65 | 10.89 | 10.23 | 16.46 | 13.12 | 14.53 |
| ours | | **10.36** | **8.90** | **8.04** | **9.32** | **10.56** | **11.96** |

Table S1. Supplementary to Table 2. **Uniform memory budget setting.** Ablation study for evaluating four exemplar methods. The top and the bottom blocks present average accuracies $\bar{\mathcal{A}}$ (%) and forgetting rates $\mathcal{F}$ (%), respectively. "*w/o* adj." means without old exemplar adjustment. In these experiments, we use ***random*** strategy to discard old exemplars, and use weight transfer operations to train MCIL models.

| Discarding Strategy | | *CIFAR-100* | | | *ImagNet-Subset* | | |
|---|---|---|---|---|---|---|---|
| | | *N*=5 | 10 | 25 | 5 | 10 | 25 |
| ***random*** | | **65.01** | **63.80** | **64.42** | **72.82** | **72.20** | **72.31** |
| ***low loss*** | ↑ | 62.76 | 62.96 | 61.31 | 70.21 | 70.63 | 69.96 |
| ***high loss*** | | 63.76 | 63.16 | 62.03 | 71.63 | 71.32 | 70.87 |
| ***random*** | | 10.36 | **8.90** | **8.04** | **9.32** | **10.56** | **11.96** |
| ***low loss*** | ↓ | 15.36 | 14.76 | 16.27 | 14.92 | 15.39 | 15.40 |
| ***high loss*** | | **9.96** | 13.82 | 13.68 | 10.02 | 14.32 | 13.87 |

Table S2. **Uniform memory budget setting.** Ablation study for comparing three exemplar discarding strategies. The top and the bottom blocks present average accuracies $\bar{\mathcal{A}}$ (%) and forgetting rates $\mathcal{F}$ (%), respectively. ***random*** performs the best in the most of cases. Note that the weight transfer operations are applied in all these experiments.

In this section, we show the results of **Uniform memory budget setting**. Table S5 presents the comparisons to related works [1, 2, 5, 8], with and without our *mnemonics* training as plug-in module. Figure S1 demonstrates the phase-wise results of our best model. Table S1 shows the ablation study that evaluates two key components: training *mnemonics* exemplars; and adjusting old *mnemonics* exemplars. Table S2 shows the ablation study of using different old exemplar discarding strategies.

## D. Ablation study for distillation loss

This is supplementary to Section 5.2 "**ablation study**". In Table S3, we supplement the ablation study of distillation loss [2, 5].

| Exemplar | | *CIFAR-100* | | | *ImagNet-Subset* | | |
|---|---|---|---|---|---|---|---|
| | | $N$=5 | 10 | 25 | 5 | 10 | 25 |
| *w/o* distill. | ↑ | 50.86 | 50.40 | 49.69 | 65.74 | 64.63 | 65.21 |
| *w/* distill. | | **64.95** | **63.26** | **63.70** | **73.30** | **72.17** | **71.50** |
| *w/o* distill. | ↓ | 38.70 | 36.94 | 34.38 | 29.32 | 27.44 | 29.64 |
| *w/* distill. | | **11.64** | **10.90** | **9.96** | **10.20** | **9.88** | **11.76** |

Table S3. Supplementary to Table 2. Ablation study for distillation loss. The top and the bottom blocks present average accuracies $\bar{\mathcal{A}}$ (%) and forgetting rates $\mathcal{F}$ (%) of using LUCIR [1] with our *mnemonics* training approach or [6] as a plug-in module. "*w/*" and "*w/o* distill." mean with and without distillation loss, respectively. Note that the weight transfer operations are applied in all these experiments.

## E. Comparison with generative replay method

In Table S4, we supplement the average accuracies $\bar{\mathcal{A}}$ (%) and forgetting rates $\mathcal{F}$ (%) of using related methods [1] with our *mnemonics* training approach or [6] as a plug-in module. For fair comparison, we implement [6] in our benchmark protocol.

| Exemplar | | *CIFAR-100* | | |
|---|---|---|---|---|
| | | $N$=5 | 10 | 25 |
| LUCIR *w/* [6] | ↑ | 64.01 | 61.23 | 58.36 |
| LUCIR *w/* ours | | **64.95** | **63.26** | **63.70** |
| LUCIR *w/* [6] | ↓ | 16.23 | 19.63 | 25.51 |
| LUCIR *w/* ours | | **11.64** | **10.90** | **9.96** |

Table S4. Average accuracies $\bar{\mathcal{A}}$ (%) (top block) and forgetting rates $\mathcal{F}$ (%) (bottom block) of using LUCIR [1] with our *mnemonics* training approach or [6] as a plug-in module. Note that the weight transfer operations are applied in "*w/* ours" methods.

## F. More visualization results

This is supplementary to Section 5.2 "**visualization results.**". In Figure S2, we supplement the changes of average distances between exemplars and initial samples. More visualization results are in the attached video and our project page: https://mnemonics.yyliu.net.

## References

[1] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 2, 3, 4

[2] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 2, 3, 4

[3] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 5

[4] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, pages 3942–3951, 2018. 1

[5] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542, 2017. 2, 3, 4

[6] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NIPS*, pages 2990–2999, 2017. 3

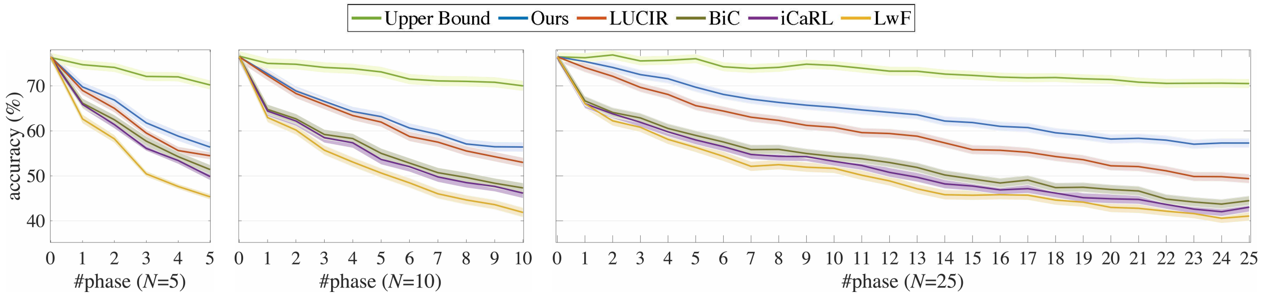
[7] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, pages 403–412, 2019. 1

[8] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 2, 3, 4

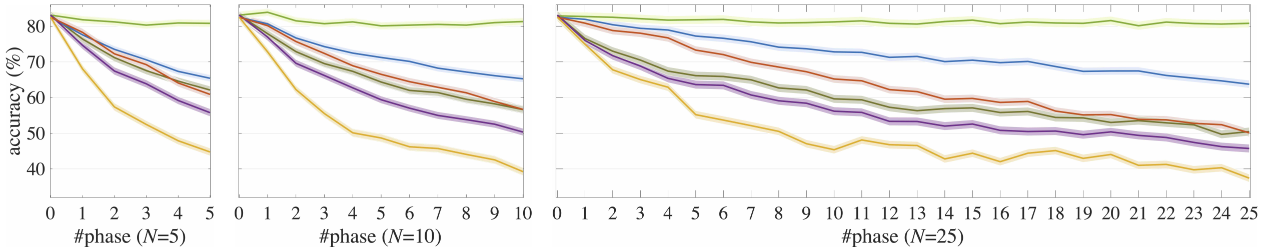
| Method | | *CIFAR-100* | | | *ImageNet-Subset* | | |
|---|---|---|---|---|---|---|---|
| | | $N$=5 | 10 | 25 | 5 | 10 | 25 |
| LwF$^\diamond$ [2] | ↑ | 56.79 | 53.05 | 50.44 | 58.83 | 53.60 | 50.16 |
| LwF *w/* ours | | 57.56 | 57.34 | 58.08 | 64.47 | 64.94 | 67.10 |
| iCaRL [5] | | 60.48 | 56.04 | 52.07 | 67.32 | 62.42 | 57.04 |
| iCaRL *w/* ours | | 60.82 | 59.28 | 58.06 | 73.83 | 72.49 | 71.40 |
| BiC [8] | | 61.35 | 56.81 | 53.42 | 70.82 | 66.62 | 60.44 |
| BiC *w/* ours | | 61.88 | 60.24 | 58.91 | 74.02 | 72.67 | 71.75 |
| LUCIR [1] | | 63.34 | 62.47 | 59.69 | 71.20 | 68.21 | 64.15 |
| LUCIR *w/* ours | | **65.01** | **63.80** | **64.42** | **72.82** | **72.20** | **72.31** |
| LwF$^\diamond$ [2] | ↓ | 41.07 | 40.20 | 38.64 | 52.48 | 53.20 | 51.16 |
| LwF *w/* ours | | 37.13 | 33.08 | 28.38 | 36.12 | 32.00 | 32.60 |
| iCaRL [5] | | 29.42 | 30.80 | 33.32 | 40.40 | 41.76 | 44.04 |
| iCaRL *w/* ours | | 25.00 | 24.44 | 24.18 | 13.24 | 16.08 | 22.80 |
| BiC [8] | | 24.96 | 27.14 | 30.66 | 25.56 | 29.40 | 35.84 |
| BiC *w/* ours | | 22.78 | 22.82 | 22.60 | 12.48 | 15.12 | 21.24 |
| LUCIR [1] | | 20.46 | 23.16 | 25.76 | 27.96 | 31.04 | 35.36 |
| LUCIR *w/* ours | | **10.36** | **8.90** | **8.04** | **9.32** | **10.56** | **11.96** |

$^\diamond$ Using *herding* exemplars as [1, 5, 8] for fair comparison.

Table S5. Supplementary to Table 1. **Uniform memory budget setting.** Average accuracies $\bar{\mathcal{A}}$ (%) (top block) and forgetting rates $\mathcal{F}$ (%) (bottom block) of using related methods [1, 2, 5, 8] with and without our *mnemonics* training approach as a plug-in module. We use ***random*** strategy to discard old exemplars, and use weight transfer operations to train MCIL models.

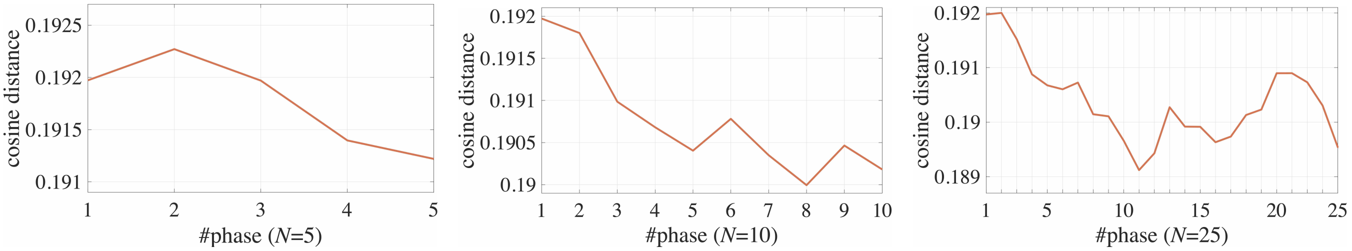


(a) CIFAR-100 (100 classes). In the 0-th phase, $\Theta_0$ is trained on 50 classes, the remaining classes are given evenly in the subsequent phases.

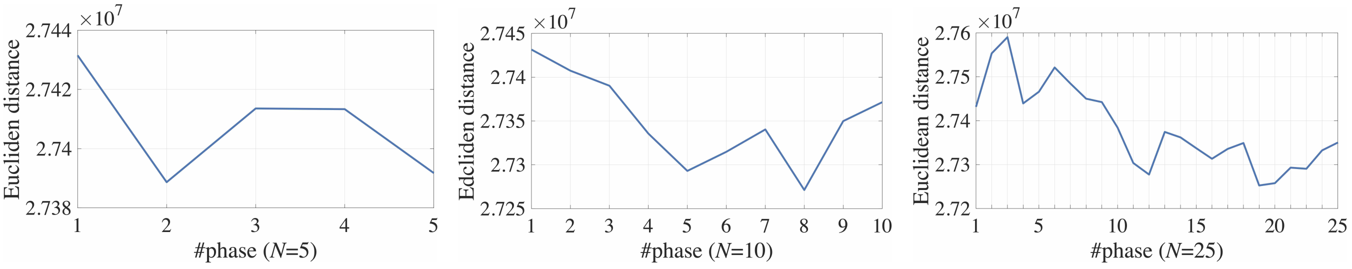


(b) ImageNet-Subset (100 classes). In the 0-th phase, $\Theta_0$ is trained on 50 classes, the remaining classes are given evenly in the subsequent phases.

Figure S1. Supplementary to Figure 4. **Uniform memory budget setting.** Phase-wise accuracies (%). Light-color ribbons are visualized to show the 95% confidence intervals. Comparing methods: Upper Bound (the results of joint training with all previous data accessible in each phase); LUCIR (2019) [1]; BiC (2019) [8]; iCaRL (2017) [5]; and LwF (2016) [2]. For ours, we show our best results using "LUCIR *w/* ours". The average accuracy of each curve is given in Table S5. Note that we apply ***random*** strategy to discard old exemplars.



(a) CIFAR-100 (100 classes). Cosine distance.



(b) CIFAR-100 (100 classes). Euclidean distance.

Figure S2. The changes of average distances between exemplars and initial samples. We show the results using "LUCIR *w/* ours". The average accuracy of each curve is given in Table 1. All curves are smoothed with a rate of 0.8 for a better visualization.

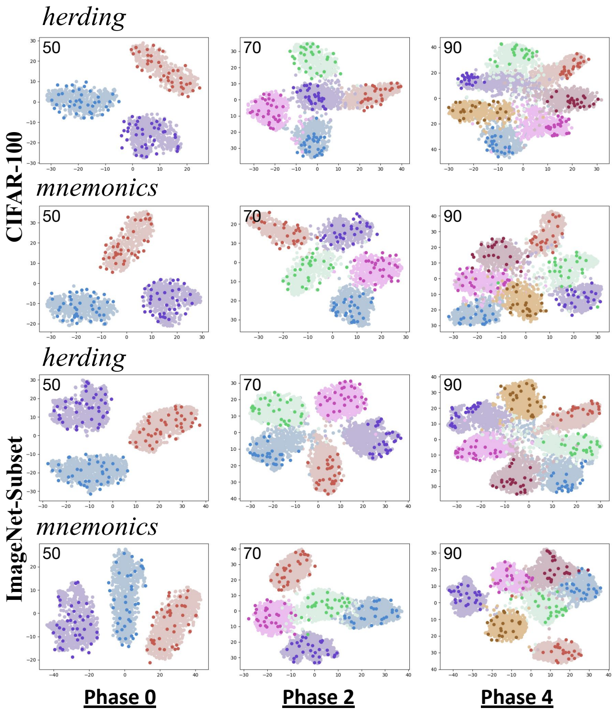


Figure S3. Supplementary to Figure 5. **Uniform memory budget setting.** The t-SNE [3] results of *herding* and our *mnemonics* on two datasets. $N = 5$. In each colored class, deep-color points are exemplars, and light-color ones show the original data as reference of the real data distribution. Total number of classes (used in real training) is given in the top-left corner of the sub-figure. For a clear visualization, Phase-0 randomly picks $3$ classes from $50$ classes on CIFAR-100 / ImageNet-Subset. Phase-2 and Phase-4 increases to $5$ and $7$ classes, respectively. Note that we apply ***random*** strategy to discard old exemplars.