Supplementary Materials

Cheng-Hsun Lei*, Yi-Hsin Chen*, Wen-Hsiao Peng, and Wei-Chen Chiu

National Chiao Tung University, Taiwan
{raygoah.cs07g, yhchen.iie07g}@nctu.edu.tw {wpeng, walon}@cs.nctu.edu.tw

1 Further Training Details

This section provides further training details. For a fair comparison, we follow the settings in iCaRL [1]. On CIFAR-100, the learning rate starts from 2.0 and is divided by 5 after 49 and 63 epochs. Moreover, 70 epochs are run in each incremental training phase. Likewise, on ImageNet the learning rate starts from 2.0 and is divided by 5 after 20, 30, 40 and 50 epochs. Each training phase has 60 epochs. On both datasets, the optimizer is SGD and the batch size is 128.

2 Rectified Cosine Normalization

This section conducts a simulation to justify our rectified cosine normalization. It is reported empirically in the main manuscript that using rectified cosine normalization for training can better encourage the separation between classes.

Specifically, in evaluating the binary cross-entropy loss $\mathcal{L}_{bce}(x_k)$, our rectified cosine normalization computes the activation $a_{i|k}$ of an image x_k for class i to be $a_{i|k} = \bar{W}_i^T \bar{F}_k$, while the cosine normalization in [2] uses $a_{i|k} = \bar{w}_i^T \bar{f}_k$. The bar indicates l_2-normalization, $W_i = (w_i, b_i)$ is formed by the concatenation of the classification weight vector w_i and an additional learnable bias b_i; likewise, $F_k = (f_k, 1)$ (referred to as the augmented feature representation) concatenates the feature f_k (referred to as the original feature representation) of the image x_k and a constant bias 1. Due to the l_2-normalization, the activation $a_{i|k}$ is in the range of $[-1, 1]$. We thus introduce a learnable η to control the curvature of the sigmoid function $\sigma(\cdot)$:

$$\sigma(a_{i|k}) = \frac{1}{1 + exp(-\eta a_{i|k})}. \tag{1}$$

Fig. 1 illustrates the benefits of our rectified cosine normalization, taking the 2-dimensional cases as examples. In these examples, each class has only one trainable data f_k (or F_k) and the corresponding class embedding w_k (or W_k), where $f_k, w_k \in R^2$ and $F_k, W_k \in R^3$. They are learned via the steepest descent algorithm. Notably, f_k (or F_k) is considered a parameter to be solved, with the aim of simulating the ideal case where the feature extractor has enough capacity.

* Both authors contributed equally to the paper

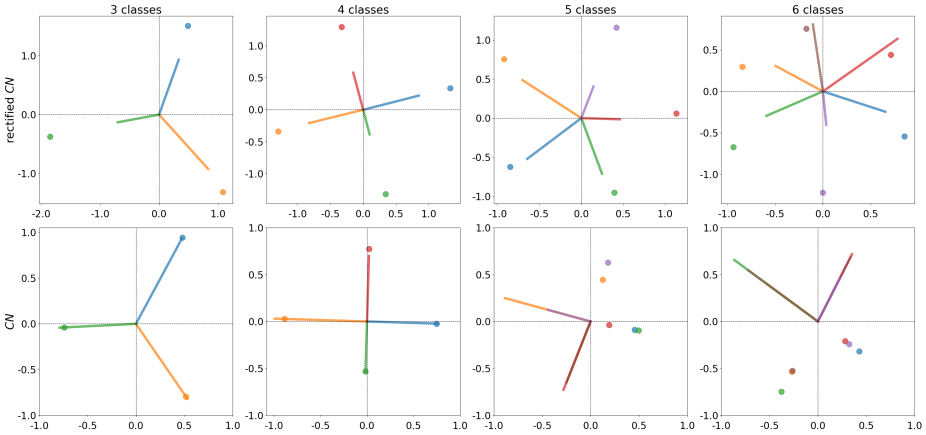


Fig. 1. Comparison of our rectified cosine normalization (denoted as rectified-CN) and the typical cosine normalization (denoted as CN) with different number of classes. Dots in various colors represent the learned feature vector f_k for class k, and the line segment of the same color denotes the corresponding weight vector w_k.

The first row presents the results of training with rectified cosine normalization in the augmented feature space $F_k = (f_k, 1)$ with $f_k = (x, y) \in R^2$. Both the learned features F_k and the class embeddings $W_i = (w_i, b_i)$ are projected onto the xy-plane for visualization and comparison. The second row corresponds to the results of using typical cosine normalization, for which both f_k and w_i are learned directly in the original feature space without augmentation. It is clear to see that in the cases with 3 and 4 classes, both rectified cosine normalization and typical cosine normalization work well. The features f_k point in the same direction as their class embeddings w_k, as expected. We note that minimizing the $\mathcal{L}_{bce}(x_k)$ (equation (6) in the main manuscript) requires that (1) F_k should be separated from the class embeddings W_i not of the same class (i.e. $i \neq k$) by at least 90 degrees and that (2) the angle between F_k and W_k of the same class should be smaller than 90 degrees and preferably be 0 degrees. These apply to f_k and w_i learned directly in the original feature space. When the number of classes is small, these requirements are easy to fulfill, be it in the 2-dimensional original feature space or in the 3-dimensional augmented feature space. They however become difficult to hold simultaneously in the 2-dimensional feature space when the number of classes increases beyond 4. Intriguingly, in those cases, the feature vectors of different classes may collapse into few modes, putting much emphasis on the first requirement due mainly to the excessive amount of negative examples (i.e. for a given f_k, the number of class embeddings $w_i, i \neq k$ not of the same class is much larger than the case $i = k$). In contrast, both requirements can still hold in the 3-dimensional augmented feature space. We observe that this is achieved by setting all b_i's to negative values and having F_k and W_k (respectively, $W_i, i \neq k$) point in approximately similar directions (respectively, in significantly different directions) when projected onto the xy-plane. The larger

degree of freedom in adapting the class embeddings W_i to the two requirements in the higher dimensional augmented feature space explains the generally better separation between feature vectors f_k with our rectified cosine normalization in the lower dimensional original feature space.

3 Detail Formulation of Our Evaluation

We follow the common metrics in prior works [1,3,2] to evaluate the performance of incremental learning; that is, incremental accuracy, average incremental accuracy. Furthermore, we additionally propose a metric called *phase accuracy*. The details of these metrics are described as follows.

Incremental Accuracy. Let $A_{j,i}$ be the average accuracy of classes learned in phase i after training the network sequentially to phase j, where $i < j$. Incremental accuracy is defined as $A_j = \sum_{i=1}^{j} A_{j,i}$, which is the accuracy for classifying all the seen classes at the end of training phase j. It is the most commonly-used metric, but has the limitation of showing only the accuracy over the seen classes as a whole without giving any detail of how the model performs on separate groups of classes (learned incrementally in each phase).

Average Incremental Accuracy. Average incremental accuracy accumulates the incremental accuracy, obtained from each training phase, up to the current phase j and then takes the average, i.e. $\frac{1}{j}\sum_{i=1}^{j} A_i$.

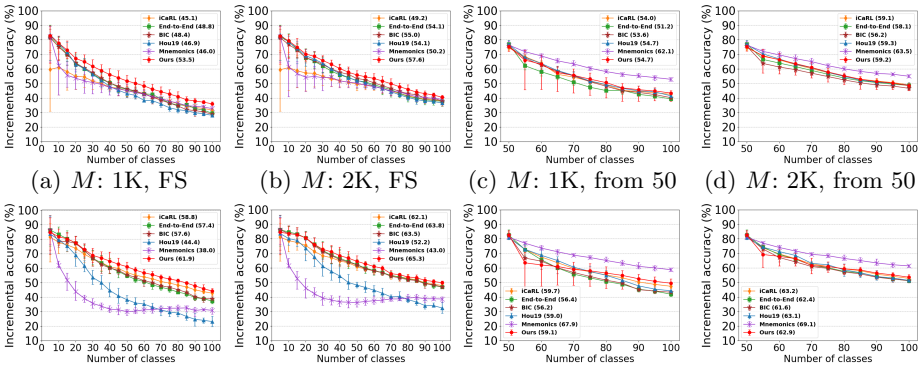
Phase Accuracy. A model that learns well incrementally should present a balanced distribution of $A_{j,i}, i \in \{1, ..., j\}$. Thus, we propose phase accuracy to evaluate at the end of the entire incremental training to present the classification accuracy on separate groups of classes (a group represents a phase in this thesis). It provides a breakdown look at whether the model would favor some groups of classes over the others as a result of catastrophic forgetting.

4 Adding 5 New Classes in Each Phase

Here we conduct more experiments based on almost the same setting as the one in the main manuscript (i.e. CIFAR-100 and ImageNet datasets, two training scenarios, memory size of 1000 and 2000, as well as the three evaluation metrics), but now the number of new classes in each incremental phase is set to 5 (e.g. in total 20 phases while training from scratch).

4.1 Incremental Accuracy Comparison

The quantitative evaluation results in terms of incremental accuracy are provided in Fig. 2, while Table 1 particularly summarizes the results at the end of the entire incremental-training. Similar to the observations we have in the main manuscript, on CIFAR-100 and ImageNet, when learning from scratch, our model outperforms all the baselines with memory size 1000 and 2000. Also, we



(a) M: 1K, FS (b) M: 2K, FS (c) M: 1K, from 50 (d) M: 2K, from 50

(e) M: 1K, FS (f) M: 2K, FS (g) M: 1K, from 50 (h) M: 2K, from 50

Fig. 2. Incremental accuracy on CIFAR-100 (top row) and ImageNet (bottom row), for memory sizes $M = 1K, 2K$ and training scenarios "FS" and "from 50". Average incremental accuracy of each method is shown in parentheses.

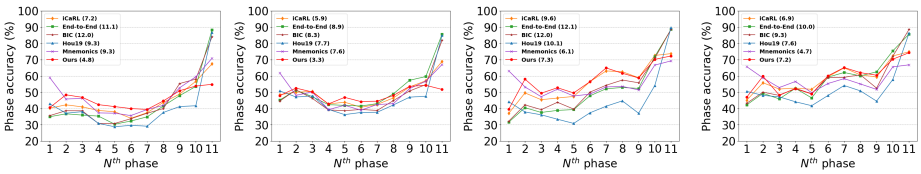
Table 1. Comparison of incremental accuracy at the end of training

Training scenario	from scratch				from 50 classes			
Dataset	CIFAR		ImageNet		CIFAR		ImageNet	
Memory size	1K	2K	1K	2K	1K	2K	1K	2K
iCaRL [1]	31.8	37.7	42.7	47.9	42.5	49.4	47.4	52.8
End-to-End [3]	29.6	37.7	37.2	46.8	39.4	48.6	41.9	51.1
BIC [4]	28.8	37.8	38.9	46.9	40.2	46.6	43.3	51.2
Hou19 [2]	28.1	36.5	23.2	32.4	41.5	48.8	44.2	51.4
Mnemonics [5]	32.7	39.1	30.9	38.6	**52.8**	**55.0**	**58.7**	**61.4**
Ours	**36.1**	**40.5**	**44.2**	**49.8**	43.4	48.4	49.5	53.6

find again that Hou19 [2] and Mnemonics [5] are unable to perform well when the feature extractor is learned from scratch. When starting from 50 classes, ours outperforms all the other baselines, except Mnemonics [5], on ImageNet. On CIFAR-100, our model performs comparably to the other baselines with memory size 2000 and achieves the second highest incremental accuracy with memory size 1000.

4.2 Phase Accuracy Comparison

Fig. 3 presents the phase accuracy for different methods with the training scenario "from 50", where the baselines perform more closely to our method in terms of incremental accuracy. Shown in parentheses is the mean absolute deviation from the average of each method's phase accuracy. The smaller the deviation, the more balanced the classification accuracy is on different classes. From the figure, our scheme is seen to achieve the minimum mean absolute deviation in phase accuracy on CIFAR-100 and comparable mean absolute deviation to iCaRL [1] and Mnemonics [5] on ImageNet.



(a) M: 1K, from 50 (b) M: 2K, from 50 (c) M: 1K, from 50 (d) M: 2K, from 50

Fig. 3. Phase accuracy comparison: (a)(b) are results on CIFAR-100 and (c)(d) on ImageNet. M is the memory size, and the model is pre-trained with 50 classes. The mean absolute deviation in phase accuracy is shown in parentheses.

Table 2. Comparison of forgetting measure under the setting that learns 10 new classes in each phase.

Training scenario	from scratch				from 50 classes			
Dataset	CIFAR		ImageNet		CIFAR		ImageNet	
Memory size	1K	2K	1K	2K	1K	2K	1K	2K
iCaRL [1]	**24.6**	20.6	**22.3**	**20.6**	24.6	20.4	24.1	20.4
End-to-End [3]	41.6	32.9	39.1	31.0	38.2	29.1	37.2	28.4
BIC [4]	37.3	28.6	36.1	31.4	33.5	26.4	33.9	27.3
Hou19 [2]	27.3	**16.8**	38.7	28.8	**17.2**	**11.1**	20.4	**13.4**
Mnemonics [5]	39.9	33.1	38.4	31.7	22.8	18.9	**15.9**	14.0
Ours	25.0	22.3	24.6	23.6	17.7	16.1	18.4	17.8

5 Forgetting Measure

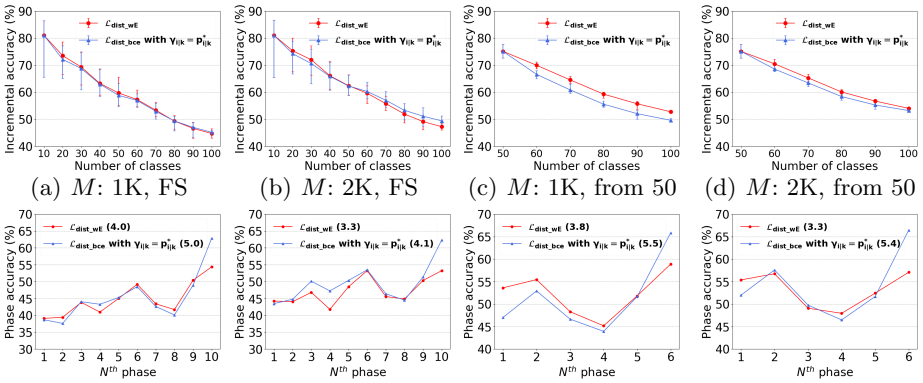
In this section, we present the results of forgetting measure [6] on every class at the end of training, in order to understand to what degree the competing methods forget the previously learned classes. Generally, the lower the forgetting measure, the less the catastrophic forgetting. Table 2 summarizes the results for the setting that learns 10 new classes in each training phase. It shows that our method has low forgetting measure and performs comparably to iCaRL [1], Hou19 [2] and Mnemonics [5].

Table 3 further provides the results for the setting of learning 5 new classes in each phase. In this setting, the model needs to be updated more frequently and tends to forget more easily what it has learned. On CIFAR-100, ours achieves the lowest forgetting measure, when learning from scratch. Moreover, it has the second lowest forgetting measure with the training scenario "from 50". On ImageNet, iCaRL [1] performs comparably to ours, when learning from scratch. Since Hou19 [2] and Mnemonics [5] are unable to perform well when learning from scratch, they suffer from serious catastrophic forgetting, showing large forgetting measure. When learning from 50 classes, ours performs comparably to Mnemonics [5] and outperforms the other baselines.

To sum up, our method has the ability to preserve well the knowledge of old classes under various settings, while showing high incremental accuracy in learning new classes and more balanced phase accuracy.

Table 3. Comparison of forgetting measure under the setting that learns 5 new classes in each phase

Training scenario	from scratch				from 50 classes			
Dataset	CIFAR		ImageNet		CIFAR		ImageNet	
Memory size	1K	2K	1K	2K	1K	2K	1K	2K
iCaRL [1]	34.5	29.4	**34.1**	**30.6**	32.7	26.7	36.3	30.5
End-to-End [3]	55.6	44.2	56.1	43.8	47.4	36.2	49.4	37.9
BIC [4]	50.4	40.9	54.6	45.7	43.2	35.9	48.5	39.9
Hou19 [2]	61.7	51.6	72.5	63.0	46.6	38.2	50.3	41.8
Mnemonics [5]	47.4	39.4	49.8	39.5	**24.8**	**20.0**	**20.2**	**20.3**
Ours	**28.0**	**23.9**	35.6	**30.6**	26.4	21.8	32.1	27.8



(a) M: 1K, FS (b) M: 2K, FS (c) M: 1K, from 50 (d) M: 2K, from 50

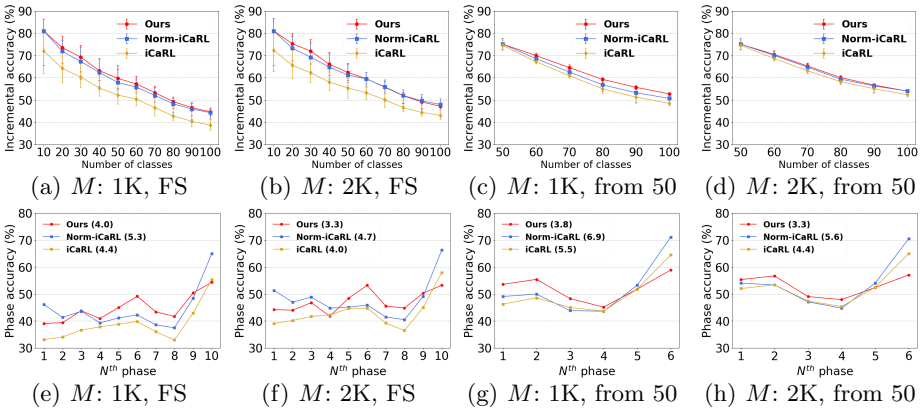
(e) M: 1K, FS (f) M: 2K, FS (g) M: 1K, from 50 (h) M: 2K, from 50

Fig. 4. Comparison of discrepancy measure $\mathcal{D}(p^*_{i|k}, p_{i|k})$ on CIFAR-100 for memory sizes $M = 1K, 2K$ and training scenario "FS" and "from 50," with the mean absolute deviation in phase accuracy shown in parentheses. The bars in the top row are evaluated by five random orderings of classes.

6 More Ablation Experiments

6.1 Symmetric vs. Asymmetric Discrepancy Measure $\mathcal{D}(p^*_{i|k}, p_{i|k})$

We argue in Sec. 3.1 of the main manuscript that an $a^*_{i|k}$-symmetric discrepancy measure is desirable. That is, $\mathcal{D}(p^*_{i|k}, p_{i|k})$ is preferably symmetric with respect to $a^*_{i|k}$ when viewed as a function of $a_{i|k}$. To single out its benefits, we change our distillation loss $\mathcal{L}_{dist\_wE}$ by replacing the $\mathcal{D}(p^*_{i|k}, p_{i|k})$ in Eq. (3) with the binary cross-entropy (BCE) between $p^*_{i|k}$ and $p_{i|k}$, and setting $\gamma_{i|k} = p^*_{i|k}$, while keeping the other design aspects (including BCE as classification loss and rectified cosine normalization) untouched. Note that these changes lead to a scheme (denoted collectively as $\mathcal{L}_{dist\_bce}$ with our $\gamma_{i|k}$) that differs from ours only in adopting an asymmetric discrepancy measure with respect to $a^*_{i|k}$. From the top row of Fig. 4, it is interesting to see that the replacement of $\mathcal{D}(p^*_{i|k}, p_{i|k}) = (\log p^*_{i|k} - \log p_{i|k})^2$ with BCE does not show much impact on incremental accuracy. However, our $a^*_{i|k}$-symmetric discrepancy measure presents more balanced phase accuracy,



(a) M: 1K, FS (b) M: 2K, FS (c) M: 1K, from 50 (d) M: 2K, from 50

(e) M: 1K, FS (f) M: 2K, FS (g) M: 1K, from 50 (h) M: 2K, from 50

Fig. 5. Comparison of iCaRL with/without rectified cosine normalization and our method on CIFAR-100 for memory sizes $M = 1K, 2K$ and training scenarios "FS" and "from 50," with the mean absolute deviation in phase accuracy shown in parentheses. The bars in the top row are evaluated by five random orderings of classes.

achieving smaller mean absolute deviations (cp. the bottom row of Fig. 4). This highlights the fact that it allows our model to strike a better balance between incremental accuracy and phase accuracy.

6.2 iCaRL with vs. without Rectified Cosine Normalization

This section investigates whether our rectified cosine normalization could also benefit iCaRL [1]. Compared in Fig. 5 are iCaRL [1] with (denoted as Norm-iCaRL) and without (denoted as iCaRL) rectified cosine normalization. Note that the other design aspects of iCaRL remain the same. As a benchmark, our scheme is also presented. A comparison between Norm-iCaRL and iCaRL in the top row of Fig. 5 reveals that rectified cosine normalization does help improve the incremental accuracy of iCaRL [1], reducing largely the performance gap between ours and iCaRL [1]. However, without our weighted-Euclidean regularization, both Norm-iCaRL and iCaRL [1] exhibit higher variations in phase accuracy than ours (cp. the bottom row of Fig. 5).

References

1. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
2. Hou, S., Pan, X., Loy, C.C., Wang, Z., Lin, D.: Learning a unified classifier incrementally via rebalancing. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
3. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: European Conference on Computer Vision (ECCV). (2018)
4. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
5. Liu, Y., Su, Y., Liu, A.A., Schiele, B., Sun, Q.: Mnemonics training: Multi-class incremental learning without forgetting. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2020)
6. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: European Conference on Computer Vision (ECCV). (2018)