APPENDIX

5.1. Training Details

ImageNet → Scenes → CUB → Flowers we use images downsampled and cropped at 64×64 (224×224 size results are shown in Sec. 5.2) in this task sequence. We train a ResNet-18 [16], using AdamW [29] optimizer with a learning rate (LR) of $1e{-}5$ and a weight decay of $5e{-}4$. We first train the model on ImageNet via (1) CE loss, or (2) SupCon loss for 500 epochs. Next, for each task in the sequence, we train the model via early stopping [6], and report the observed accuracy over the test set (see Tab. 1).

The LP classifiers are trained using the same optimizer and weight decay value. However, for faster convergence, in this setting, we use a cosine LR scheduler [28] with an initial LR of $1e{-}4$. The results of LP classifiers are reported on the ImageNet validation set. All the training images in this section undergo, random crop, random horizontal flip, and color jitter of 0.5.

SplitCIFAR100, MiniImageNet, ImageNet32 For our SplitCIFAR100 10-task sequence, MiniImageNet 20-task sequence, and ImageNet32 200-task sequence we use SGD optimizer with LR of 0.05, momentum of 0.9, and weight decay of $1e{-}4$. We train the models for 50 epochs for SplitCIFAR100 and MiniImageNet, and for 80 epochs for ImageNet32. The augmentation pipeline consists of random crop, random horizontal flip, and color jitter of 0.5. For the LP classifiers, we train each for 20 epochs, using AdamW [29] optimizer with a LR of $1e{-}3$ and a weight decay of $5e{-}4$.

For the SplitCIFAR10 2-task sequence from [38] we use the hyper-parameters and training conditions mentioned in [38] for both VGG [45] and ResNet [16] architectures. For the LP classifiers, we train each for 70 epochs, using AdamW [29] optimizer with a LR of $1e{-}3$ and a weight decay of $5e{-}4$. For LP training we use data augmentation in all cases, except the SplitCIFAR10 to obtain the most accurate measure of the model's ability to linearly separate the data of interest. For SplitCIFAR10 2-task we choose not to use any data augmentation because the source result ([38]) does not rely on data augmentation for training. We opt not to use data augmentation in our LP evaluation for a fair comparison.

5.2. ImageNet→ Scenes→ CUB with 224 × 224: Reproducing Li et al.

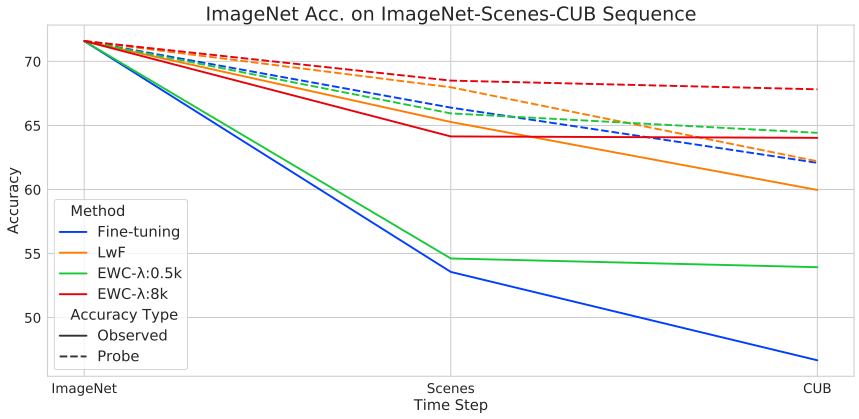

Figure 6. Performance on ImageNet during the transfer sequence (ImageNet→Scenes→CUB) using VGG-16. We observe that although observed accuracy heavily degrades, the LP accuracy for finetuning does not decay as drastically and can rival LP accuracy of methods such as LwF and EWC. We evaluate methods which do not rely on storing data from Task 1 to replay during training. Note EWC with $\lambda = 8k$ is the best performing method in terms of LP and observed accuracy, however it does not perform well on the current task (see Tab. 6).

Method	Acc. Scenes	Acc. CUB
Finetuning	74.70%	74.39%
LwF	74.78%	75.23%
EWC$_{\lambda:0.5k}$	74.70%	74.72%
EWC$_{\lambda:8k}$	72.69%	71.44%

Table 6. Observed accuracy of the current task in the sequence ImageNet→Scenes→CUB using VGG-16 architecture. Although EWC$_{\lambda:8k}$ attains relatively poor performance on the current task, it achieves the highest LP and observed accuracy for the previously seen tasks (see Fig. 6).

For ease of experiments, in Sec. 4.1 we have used a lower resolution, 64×64, a light ResNet-18 model, and a modern rapid training scheduler. On the other hand in this section we reproduce the results of the original paper (Li *et al.*) [26] using a VGG-16 model and 224×224 input size and then carry out our LP analysis further confirming our observations. We take the setting of [26], which considers the ImageNet [42] transfer to various datasets, in particular CUB [46], and Scenes [37]. We use the same model architecture (VGG-16) and training procedures described in [26], which proposes LwF, closely reproducing their ImageNet → Scenes as the first step in the sequence (see Tab. 7). We also include an EWC baseline under

two conditions: (a) large λ value ($\lambda = 8k$), so the network is inclined to preserve the knowledge important to the previous tasks, and (b) small λ value ($\lambda = 0.5k$), so the network is encouraged to perform competitively on the current task. The results are shown in Figure 6 and Table 6.

We first note that our results for the first task switch are consistent with those reported in [26] (see Tab. 8). Fig. 6 reveals that although the forgetting in terms of the traditional measure is high for finetuning compared to LwF (as shown in [26]), the LP accuracy of these methods suggest a much less drastic forgetting. Furthermore, the LP performance across finetuning and other methods is not as drastically different as their respective observed accuracies are. Indeed, we observe that on the third task, finetuning outperforms LwF in representation forgetting on ImageNet. Similarly EWC does not clearly outperform naive finetuning. For example, if using one hyper-parameter for the regularization term the performance closely tracks finetuning. On the other hand using $\lambda = 8k$ we observe the best LP performance on ImageNet through the task sequence but degraded current task performance as seen in Tab. 6.

	ImageNet (T-1) → CUB (T-2)	
	Obs. T-2 Acc.	Obs. T-1 Acc.
Finetune- [26]	73.1%	50.7%
Finetune-Ours	74.5%	50.9%
LwF- [26]	72.5%	60.6%
LwF-Ours	75.7%	63.6%
	ImageNet (T-1) → Scenes (T-2)	
Finetune- [26]	74.6%	62.7%
Finetune-Ours	74.7%	53.6%
LwF- [26]	74.9%	66.8%
LwF-Ours	74.8%	65.3%

Table 7. Reproduction of the results reported in [26]. Note that we observe a slight difference in our reproduced results due to stochasticity of training neural networks, and removing the warm-up step.

Observed Acc. on ImageNet: 71.59%			
	ImageNet (T-1) → CUB (T-2)		
	T-1 Acc. @ T-2	LP Acc. @ T-2	T-2 Acc.
FT	50.89%	64.49%	74.51%
LwF	63.58%	67.23%	75.65%
$\text{EWC}_{\lambda:8k}$	60.28%	67.46%	72.70%
$\text{EWC}_{\lambda:0.5k}$	50.78%	63.99%	74.53%
	ImageNet (T-1) → Scenes (T-2)		
FT	53.56%	66.39%	74.70%
LwF	65.27%	67.98%	74.78%
$\text{EWC}_{\lambda:8k}$	64.14%	68.50%	72.69%
$\text{EWC}_{\lambda:0.5k}$	54.61%	65.94%	74.70%

Table 8. Forgetting of Task 1 measured via optimal linear probes (LP). Note that although the forgetting is much higher for finetuning compared to LwF, the LP accuracy is nearly identical, especially for the ImageNet → Scenes task, suggesting that LwF does not improve over naive finetuning in terms of forgetting knowledge acquired on ImageNet.

Although we followed the training procedure as closely as possible to the ones reported by [26], the results are slightly different from the ones reported in [26] due to (a) not using the task-head warm-up step, where the backbone network is first frozen and the newly added task head is trained until convergence (warm-up), and then the entire network is trained until convergence, and (b) stochasticity of training neural networks. Table 7 highlights these differences.

5.3. Comparing Overall Representation Improvement

In addition to representation forgetting, we consider also measuring how much a representation improves overall as seen by a linear probe trained and evaluated on data from the union of all current and future tasks. We can evaluate this by training at each step in the sequence an "All-LP", that is a linear probe trained on all the training data and evaluated on all test data. A natural baseline to compare such an approach to is splitting iid data into 10 subsets trained in sequence (we denote this iid-split). The results of this evaluation are shown for SplitCIFAR100 in Figure 7. We observe again that SupCon exceeds LwF methods and competes with the small replay-sized ER method.

5.4. Representation Forgetting for Other Tasks

Complementing the results in Sec. 4.1, we report the observed and LP accuracies for methods when measured for tasks beyond Task 1. Specifically for SplitCIFAR100 we show trends for Task 2, 3, and 5 in Fig. 8. We observe similar trends as reported for Task 1 in Sec. 4.1.

5.5. Effect of Increased Model Capacity for SupCon

To complement the results in Sec. 4.2, we also compare finetuning with SupCon and CE in terms of its properties on wide and deep networks. Since SupCon training does not have an observed accuracy we compare only the LP performance

unlike Sec. 4.2. From Tab. 9, we observe that similar to CE the LP performance improves with depth and exceeds that of LwF [26], and nearly matching ER-M5 with greater width.

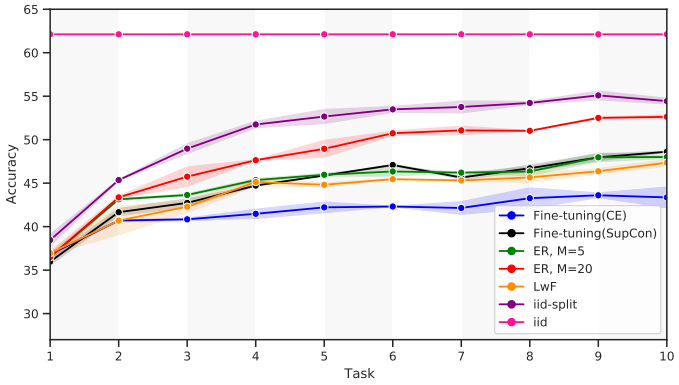

Figure 7. All-LP anytime evaluation plot on SplitCIFAR100 10-task sequence. All-LP is a probe trained on all, *i.e.* seen and unseen tasks, training data and evaluated on all test data. We compare this with splitting iid data into 10 subsets trained in sequence, denoted as iid-split.

		Task 1 LP T=10	LP Acc. All T=10
fine(SupCon)	RN18, Width=32	70.4	74.0
	RN18, Width=128	74.3	77.1
	RN101, Width=32	71.9	75.4
fine(CE)	RN18, Width=32	64.8	70.8
	RN18, Width=128	70.5	74.2
	RN101, Width=32	67.9	72.4
ER-M5	RN18, Width=32	74.2	75.7
	RN18, Width=128	75.6	77.3
	RN101, Width=32	74.5	76.1
ER-M20	RN18, Width=32	76	76.4
	RN18, Width=128	78.8	80.1
	RN101, Width=32	77.1	77.5
LwF	RN18, Width=32	70.1	73.4
	RN18, Width=128	74.8	76.7
	RN101, Width=32	71.0	74.6

Table 9. Final Accuracy of 10 task SplitCIFAR100 sequence with variable width and depth in the offline setting. M indicates the number of samples per task used in the ER buffer. We compare SupCon LP to others showing it has similar improvements in width and depth to non-finetuning methods.

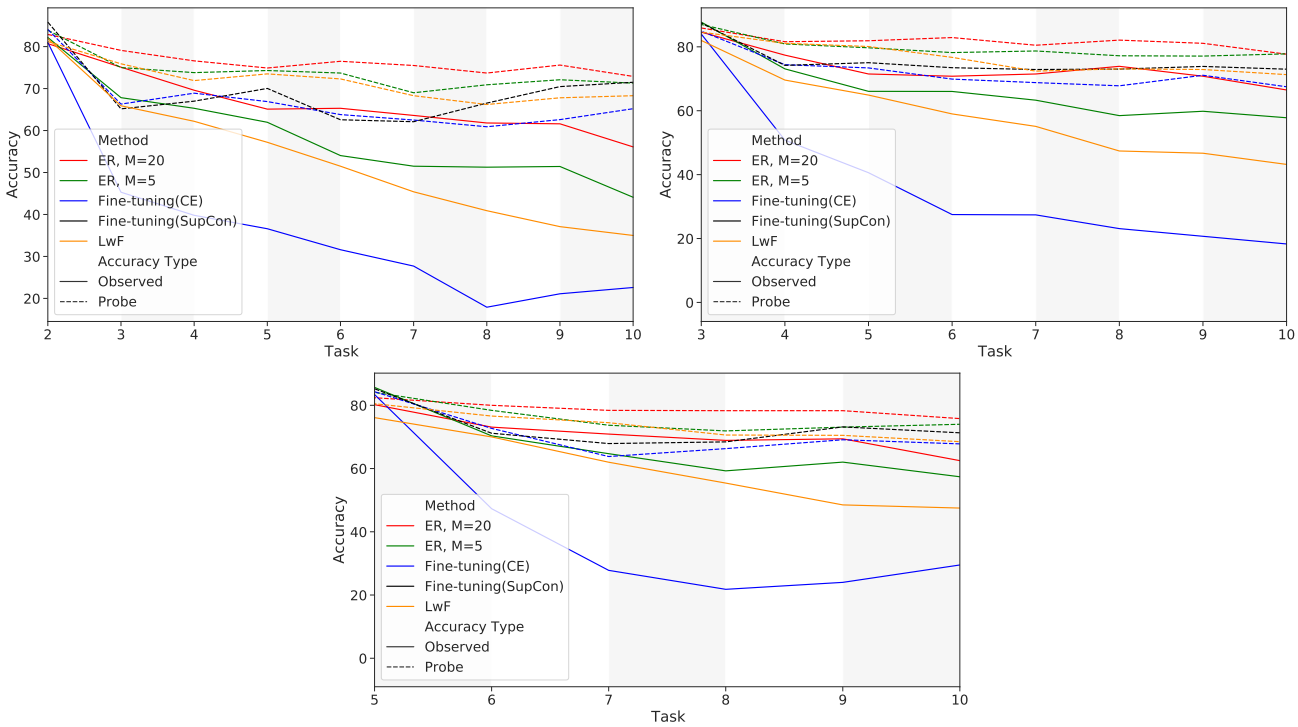

Figure 8. LP and Observed accuracy for Task 2 (upper left), 3 (upper right), and 5(bottom) on 10-Task SplitCIFAR100.