Supplementary: Essentials for Class Incremental Learning

Sudhanshu Mittal Silvio Galesso Thomas Brox
University of Freiburg, Germany
mittal,galessos,brox@cs.uni-freiburg.de

A. Experiment Details

This section includes details concerning experiments included in the paper: Adapting weighting (AW) details, hyperparameter details and standard deviation for all the results. We report all the evaluation metrics averaged over 3 run trials (unless mentioned otherwise) to capture the variance in the class-IL training process.

A.1. Adaptive Weighting (AW)

In each incremental step, training a network comprises a classification loss and a distillation loss to preserve knowledge about previous classes. Our baseline contains an adaptive weighting function λ (similar to [1]) between two losses:

$$\lambda = \lambda_{base} \left(\frac{C_n + C_o}{C_n} \right)^{2/3} \tag{1}$$

,where C_n denotes number of new classes, C_o denotes number of old classes, λ_{base} is fixed constant for each method. It dynamically increases weightage on preserving old knowledge as incremental training continues. It improves the baseline model by 0.45% for 5 task experiment on CIFAR-100. $\lambda_{base} = 5$ is set for CIFAR-100, $\lambda_{base} = 20$ for ImageNet-100 and $\lambda_{base} = 600$ for ImageNet.

A.2. Experiment Details

Dataset: CIFAR-100 classes are shuffled using a fixed seed (Numpy [2] seed:1993) across all methods for fair comparison. The ImageNet-100 dataset has 100 randomly sampled classes (using Numpy seed:1993) from ImageNet and further shuffled (using Numpy seed:1993). It contains around 128K images of size 224×224 for training and 5K images for evaluation. ImageNet-1k classes are also shuffled using a Numpy seed:1993.

Optimizer: On CIFAR-100, the base network is trained for 120 epochs using a cosine learning rate schedule, where the base learning rate is 1e-1. Subsequent N tasks are trained for 240 epochs with a base learning rate of 1e-2. The learning rate is decayed until 1e-4. We use a batch size of 100 for CIFAR-100 experiments. Networks for CIFAR-100 dataset is optimized using the SGD optimizer with a momentum of 0.9 and weight decay of 5e-4.

For ImageNet-100, the network is trained for 70 epochs using a step learning rate schedule, where the base learning rate is 1e-1 for the base task and 1e-2 for the subsequent N tasks. The base learning rate is divided by 10 at {30, 60} epochs.

For ImageNet, base task is trained for 70 epochs following a step learning rate, where the base learning is 1e-1. The base learning rate is divided by 10 at {30, 60} epochs. The incremental task is trained for 40 epochs following a step learning rate, where the base learning rate starts from 1e-2. The base learning rate is divided by 10 at {25, 35} epochs. Networks for ImageNet datasets are optimized using the SGD optimizer with a momentum of 0.9 and weight decay of 1e-4. We use a batch size of 128 for both ImageNet datasets.

A.3. Overfitting Experiment

Results with standard deviation Table 1 shows class-IL performance using average accuracy and forgetting rate, and quality of secondary information using SS-NLL and SS-Acc for each class-IL runs using increasingly overfitted model snapshots. Average incremental accuracy and forgetting rate is computed for class-IL model trained over different snapshots (every 100^{th}) from the above run. Table 2 shows expected calibration error (ECE) with standard deviation for different snapshots of the overfitted model. It shows that ECE monotonically increases with the number of training epochs. Tables includes values averaged over 5 runs with respective standard deviation.

A.4. Regularization

All the regularizers are applied at base and all incremental steps, however major improvement is observed due its usage in the initial base task.

Self-distillation In the experiments, self-distillation is conducted over 4 generations (optimized using validation

Epoch	SS-NLL ↓	SS-Acc ↑	Avg Acc ↑	$\mathcal{F}$ ↓	$\mathcal{R}_\phi$↓
100	2.54 ± 0.04	38.68 ± 0.89	65.42 ± 0.06	16.03 ± 0.36	9.04 ± 0.24
200	2.89 ± 0.06	32.88 ± 0.59	65.05 ± 0.08	16.04 ± 0.26	9.27 ± 0.42
300	3.03 ± 0.06	30.09 ± 0.53	64.72 ± 0.07	16.94 ± 0.61	9.51 ± 0.23
400	3.09 ± 0.07	29.04 ± 0.68	64.3 ± 0.12	18.38 ± 0.19	9.68 ± 0.17
500	3.11 ± 0.03	27.97 ± 0.54	62.92 ± 0.11	18.57 ± 0.39	10.00 ± 0.20

Table 1: The effect of overfitting on class-IL performance and its correlation with secondary information. Table shows the performance of the network snapshots taken at every 100^{th} epoch. Accuracy decreases and SS-NLL increases, both monotonically, as more severely overfitted models are evaluated. Forgetting rate $\mathcal{F}$ also correlates with overfitting. Results are computed over 5 runs.

Epoch	ECE
100	0.093±0.003
200	0.118±0.003
300	0.126±0.004
400	0.131±0.005
500	0.137±0.002

Table 2: Expected Calibration Error for different snapshots (every 100^{th} epoch) of the overfitted model.

performance) for CIFAR-100 and ImageNet-100 dataset, and over 2 generations for ImageNet dataset. In the beginning of each self-distillation generation, the network snapshot (student) becomes the teacher network and the student continues to train (fine-tuned) with a combination of classification and distillation loss.

For CIFAR-100 experiments, the first base model is trained for 120 epochs following a cosine learning rate schedule, decaying from a learning rate 1e-1 to 1e-4. For self-distillation generations, the model is trained for 70 epochs with a decaying (cosine) learning rate from 1e-1 to 1e-3. All other optimizer settings are the same as the baseline model.

For ImageNet-100 experiments, first base model is trained for 70 epochs following a step learning rate schedule. For self-distillation generations, the model is trained for 30 epochs each where base learning rate is 1e-2 and it is divided by 10 at 10, 20 epochs.

For ImageNet experiments, the first base model is trained for 70 epochs following a step learning rate schedule. For self-distillation generations, the model is trained for 15 epochs each where base learning rate is 1e-2 and it is divided by 10 at 8, 12 epochs.

Model	**SS Metrics** (5 tasks)	
	SS-NLL ↓	SS-Acc. ↑
CCIL	2.784 ± 0.014	34.83 ± 0.654
CCIL + SD	2.675 ± 0.037	37.26 ± 0.251
CCIL + H-Aug	2.051 ± 0.013	47.69 ± 0.590
CCIL + LS	3.103 ± 0.013	24.25 ± 0.278
CCIL + Mixup	2.791 ± 0.006	31.57 ± 0.256

Table 3: Effect of regularization on secondary information. All the metrics are evaluated on the network trained on the first task. Values that are better than the baseline CCIL method are marked in green whereas the worse ones are marked in red. SD:self-distillation, LS:label-smoothing.

Results with standard deviations Table 3 shows the effect of different regularization on the quality of secondary class information. Table 4 shows the effect of different regularization on class-IL performance in terms of average incremental accuracy and forgetting rate. All experiments are conducted on CIFAR-100 dataset.

B. Representations: Qualitative Analysis

This section provides a qualitative analysis on the effect of different regularizers on the feature representations (penultimate-layer activations). We analyze the representations of the network trained on 50 classes (first task) of CIFAR-100 dataset using ResNet-32 network.

B.1. Class-mean Representations

We argue that the classes which are semantically similar must be closer in the representation space as compared to the dissimilar classes since they share more features. Based on this argument we analyze the effect of different regularization methods on the relative distances between class-mean representations. We utilize the fine- and coarse-label structure of the CIFAR-100 dataset to compare the effect on the distance between semantically similar and dissimilar classes relative to the default baseline model. Classes associated with the same coarse label or superclass are considered as similar classes, whereas dissimilar classes are picked from different superclasses. L2 distance is used as the distance metric.

Figure 1 show this qualitative analysis for two classes: *cup* and *tulip*. For example cup and can are semantically

Model	Avg. Acc. ↑		Forgetting (5 tasks)	Retention
	5 tasks	10 tasks	$\mathcal{F}\downarrow$	$\mathcal{R}_\phi\downarrow$
CCIL	66.44 ± 0.31	64.86 ± 0.40	17.13 ± 1.12	9.70 ± 0.15
CCIL + SD	67.17 ± 0.14	65.86 ± 0.29	16.81 ± 0.25	8.88 ± 0.35
CCIL + H-Aug	71.66 ± 0.23	69.88 ± 0.36	13.37 ± 0.60	6.73 ± 0.45
CCIL + LS	63.08 ± 0.21	61.99 ± 0.30	18.79 ± 0.29	12.83 ± 0.41
CCIL + Mixup	62.31 ± 0.46	57.75 ± 1.64	24.56 ± 2.52	16.01 ± 0.16

Table 4: Effect of regularization on class-IL performance. All the metrics are evaluated on the network trained on the first task. ↓ and ↑ in the column headings indicate that lower and higher values are better respectively. Values that are better than our baseline method (CCIL) are marked in green whereas the worse ones are marked in red. SD:self-distillation, LS:label-smoothing.

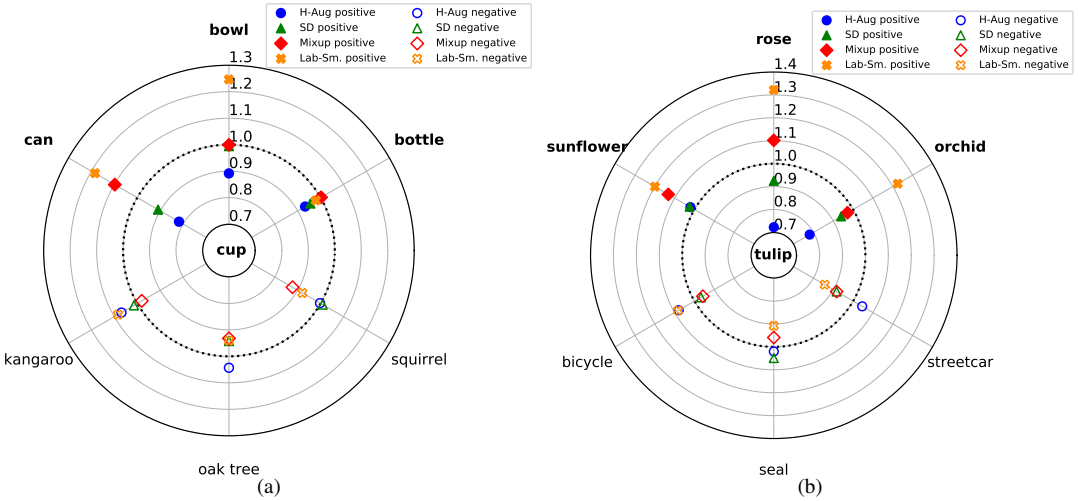

Figure 1: Effect of regularizers on the distance between mean class representations. The numbers shown in the plot are the ratios between the class means distances of each method and of the default CCIL model. Similar classes are marked in **bold**. *Dotted* circle at 1.0 depicts distances between classes in the baseline CCIL model and other distances are depicted relative to the baseline model. *Positive* and *negative* cases indicate similar and dissimilar classes respectively.

similar classes. When self-distillation and augmentation are used as regularizers, the relative distance reduces to 0.9 and 0.8 respectively, whereas when label-smoothing and mixup are applied, the relative distance increases to 1.2 and 1.1 respectively. Other similar classes follow a similar trend, whereas dissimilar pairs show an opposite behavior. Overall we find that regularizers: self-distillation and heavy data-augmentation reduce the relative distance between the similar classes (marked in bold) while not affecting or increasing distance between dissimilar classes. Whereas mixup and label smoothing increase the relative distance between similar classes and reduce the relative distance between dissimilar classes. We notice that these observations agree with the findings on secondary class information presented in the main paper.

Earlier in the main paper, we argued that label-smoothing and mixup regularization deteriorate secondary class information since they dismantle the natural output distribution. This qualitative analysis supports our argument showing how they conversely hamper the distances between similar and dissimilar classes.

References

[1] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[2] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 2011. 1