[Appendix]
Continual Learning on Noisy Data Streams via Self-Purified Replay

This appendix enlists the following additional materials.

I. Posterior in the Beta Mixture Model. Sec. 1

II. Extended Related Work. Sec. 2

III. Experiment Details. Sec. 3

IV. Extended Results & Analyses. Sec. 4

- i. Efficiency of Eigenvector Centrality. Sec. 4.1
- ii. Noise-Free Performance. Sec. 4.2
- iii. Noise Robustness Comparison. Sec. 4.3
- iv. Features from ImageNet Pretrained Model. Sec. 4.4
- v. Analyses of Stochastic Ensemble Size (E_{max}). Sec. 4.5
- vi. Filtering performances on CIFAR-100. Sec. 4.6
- vii. Self-Replay with Noisy Labeled Data. Sec. 4.7
- viii. Analyses of the Results on WebVision. Sec. 4.8
- ix. Episode Robustness. Sec. 4.9
- x. Buffer Size Analysis. Sec. 4.10
- xi. Variance. Sec. 4.11

1. Posterior in the Beta Mixture Model

We provide some details about how to fit beta mixture models [26] with the EM-algorithm [12] to obtain the posterior $p(z|c)$ for the central point with score c.

In the E-step, fixing π_z, α_z, β_z, we update the latent variables using the Bayes rule:

$$\gamma_z(c) = p(z|c) = \frac{\pi_z p(c|\alpha_z, \beta_z)}{\sum_{j=1}^{Z} \pi_j p(c|\alpha_j, \beta_j)}. \tag{1}$$

In the M-step, fixing the posterior $\gamma_z(c)$, we estimate the distribution parameters α and β using method of moments:

$$\alpha_z = \bar{c}_z\left(\frac{\bar{c}_z(1-\bar{c}_z)}{s_z^2} - 1\right), \quad \beta_z = \frac{\alpha_z(1-\bar{c}_z)}{\bar{c}_z}, \tag{2}$$

where $\bar{c}_z$ is the weighted average of the centrality scores from all the points in the delayed batch, and s_z^2 is the weighted variance estimate as

$$\bar{c}_z = \frac{\sum_{i=1}^{N} \gamma_z(c_i) c_i}{\sum_{i=1}^{N} \gamma_z(c_i)}, \tag{3}$$

$$s_z^2 = \frac{\sum_{i=1}^{N} \gamma_z(c_i)(c_i - \bar{c}_z)^2}{\sum_{i=1}^{N} \gamma_z(c_i)}, \tag{4}$$

$$\pi_z = \frac{1}{N} \sum_{i=1}^{N} \gamma_z(c_i). \tag{5}$$

Finally, we arrive at $p(z|c) \propto p(z)p(c|z)$.

2. Extended Related Work

2.1. Continual Learning

Continual learning is mainly tackled from three main branches of regularization, expansion, and replay.

Regularization-based Approaches. Methods in this branch prevent forgetting by penalizing severe drift of model parameters. Learning without Forgetting [42] employs knowledge distillation to preserve the previously learned knowledge. Similarly, MC-OCL [14] proposes batch-level distillation to balance stability and plasticity in an online manner. Elastic Weight Consolidation [32] finds the critical parameters for each task by applying the Fisher information matrix. Recently, Selfless Sequential Learning [2] enforces representational sparsity, reserving the space for future tasks.

Expansion-based Approaches. Many methods in this branch explicitly constrain the learned parameters by freezing the model and instead allocate additional resources to learn new tasks. Progressive Neural Network [59] prevent forgetting by prohibiting any updates on previously learned parameters while allocating new parameters for the training of the future tasks. Dynamically Expandable Networks[73] decides on the number of additional neurons for learning new tasks using $L2$ regularization for sparse and selective retraining. CN-DPM [36] adopts the Bayesian nonparametric framework to expand the model in an online manner.

Replay-based Approaches. The replay-based branch maintains a fixed-sized memory to rehearse back to the model to mitigate forgetting. The fixed-sized memory could be in the form of a buffer for the data samples of previous tasks or the form of generative model weights [60] to generate the previous tasks' data. GEM [43] and AGEM [6] use a buffer to constrain the gradients in order to alleviate forgetting. In [7], training a model even on tiny episodic memory can achieve an impressive performance. Some recent approaches[58, 27] combine rehearsal with meta-learning to find the balance between transfer and interference.

Online Sequential Learning. Online sequential learning is closely related to continual learning research, as it assumes that a model can only observe the training samples once before discarding them. Thus, it is a fundamental problem to maintain the buffer or selecting the samples to be rehearsed. ExStream [22] proposes the buffer maintenance method by clustering the data in an online manner. GSS [55] formulates sample selection for the buffer as a constraint reduction, while MIR [1] proposes a sample retrieving method from the buffer by selecting the most interfered samples. Considering real-world data are often imbalanced and multi-labeled, PRS [30] tackles this problem by partitioning the buffer for each class and maintaining it to be balanced. Also, combining graphs or meta-learning with online continual learning has been studied. Graphs are adopted to represent the relational structures between samples [64], and the meta-loss is applied for learning not only model weights but also per-parameter learning rates [18]. Recently, GDumb [54] and MBPA++ [11] show training a model at inference time improves the overall performance.

2.2. Noisy Labels

Learning with noisy labeled data has been a long-studied problem. In several works [76, 4, 46] make an important empirical observation that DNNs usually learn the clean data first then subsequently memorize the noisy data. Recently, a new benchmark [28] has been proposed to simulate real-world label noise from the Web. Noisy labeled data learning can be categorized into loss regularization, data reweighting, label cleaning, clean sample selection via training dynamics.

Loss Regularization. This approach designs the noise correction loss so that the optimization objective is equivalent to learning with clean samples. [52] proposes using a noise transition matrix for loss correction. [17] appends a new layer to DNNs to estimate the noise transition matrix while [24] additionally uses a small set of clean data. [77] studies a set of theoretically grounded noise-robust loss functions that can be considered a generalization of the mean absolute error and categorical cross-entropy. [70, 21] propose new losses based on information theory. [37] adopts the meta-loss to find noise-robust parameters. [3] uses a bootstrapping loss based on the estimated noise distribution.

Data Re-weighting. This approach suppresses the contribution of noisy samples by re-weighting the loss. [57] utilizes meta-learning to estimate example importance with the help of a small clean data. [67] uses a Siamese network to estimate sample importance in an open-set noisy setting.

Label Cleaning. This approach aims at explicitly repairing the labels. [45] shows that using smooth labels is beneficial in noisy labeled data learning. [63, 72] propose to learn the data labels as well as the model parameters. [56, 62] relabel the samples using the model predictions. Additionally, [33] adopts the active learning strategy to choose the samples to be re-labeled. [65, 41, 15] employ multiple models, while [20, 48, 38] utilize prototypes to refine the noisy labels.

Training Procedures. Following the observations that clean data and easy patterns are learned prior to noisy data [76, 4, 46], several works propose filtering methods based on model training dynamics. [29] adopts curriculum learning by selecting small loss samples. [69, 15, 19, 74, 47] identify the clean samples using losses or predictions from multiple models and feed them into another model. [25, 53, 49] filter noisy samples based on the accumulated losses or predictions. [8] proposes to fold the training data and filter clean samples by cross-validating those split data.

2.3. Self-supervised Learning

Self-supervised learning enables the training of a model to utilize its own unlabeled inputs and often shows remarkable performance on downstream tasks. One example of self-supervised learning uses a pretext task, which trains a model by predicting the data's hidden information. Some examples include patch orderings [13, 50], image impainting [51], colorization [71], and rotations [16, 10]. Besides designing heuristic tasks for self-supervised learning, some additional works utilize the contrastive loss. [9] proposes a simpler contrastive learning method, which performs representation learning by pulling the randomly transformed samples closer while pushing them apart from the other samples within the batch. [23] formulates contrastive learning as a dictionary look-up and uses the momentum-updated encoder to build a large dictionary. Recently, [39] extends instance-wise contrastive learning to prototypical contrastive learning to encode the semantic structures within the data.

3. Experiment Details

We present the detailed hyperparameter setting of SPR training as well as the baselines. We resize the images into 28×28 for MNIST [35], 32×32 for CIFAR-10 [34], and 84×84 for WebVision [40]. We set the size of delayed and purified buffer to 300 for MNIST, 500 for CIFAR-10,

and 1000 for WebVision on all methods. We use the batch size of self-supervised learning as 300 for MNIST, 500 for CIFAR-10, and 1000 on WebVision. The batch size of supervised learning is fixed to 16 for all experiments. The number of training epochs for the base and expert network are respectively 3000 and 4000 on all datasets, while finetuning epochs for the inference network is 50. The NTXent loss [9] uses a temperature of 0.5, and $E_{max} = 5$ for SPR. We use the Adam optimizer [31] with setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.0002$ for self-supervised training of both base and expert network, and $\epsilon = 0.002$ for supervised finetuning.

The hyperparameters for the baselines are as follows.

- Multitask [5]: We perform i.i.d offline training for 50 epochs with uniformly sampled mini-batches.
- Finetune: We run online training through the sequence of tasks.
- GDumb [54]: As an advantage to GDumb, we allow CutMix [75] with $p = 0.5$ and $\alpha = 1.0$. We use the SGDR [44] schedule with $T_0 = 1$ and $T_{mult} = 2$. Since access to a validation data in task-free continual learning is not natural, the number of epochs is set to 100 for MNIST and CIFAR-10 and 500 for WebVision.
- PRS [30]: We set $\rho = 0$.
- L2R [57]: We use meta update with $\alpha = 1$, and set the number of clean data per class as 100 and the clean update batch size as 100.
- Pencil [72]: We use $\alpha = 0.4$, $\beta = 0.1$, stage1 = 70, stage2 = 200, $\lambda = 600$.
- SL [68]: We use $\alpha = 1.0$, $\beta = 1.0$.
- JoCoR [69]: We set $\lambda = 0.1$.
- AUM [53]: We set the learning rate to 0.1, momentum to 0.9, weight decay to 0.0001 with a batch size of 64 for 150 epochs. We apply random crop and random horizontal flip for input augmentation.
- INCV [8]: We set the learning rate to 0.001, weight decay to 0.0001, a batch size 128 with 4 iterations for 200 epochs. We apply random crop and random horizontal flip for input augmentation.

4. Extended Results & Analyses

We provide more in-depth results and analyses of the experiments in this section.

4.1. Efficiency of Eigenvector Centrality

The time and space complexity of Eigenvector centrality is $O(n^2)$, where n is the number of data. Our online scenario constraints the size of n (Delayed buffer size) to be less than 2% of the entire dataset. Also, for k classes, the complexity reduces to $O((n/k)^2)$ since the Self-Centered filter computes per class. On Quadro RTX GPU, building the adjacency matrices took less than $0.0003s$. On a CPU, Eigenvector centrality computation took $0.4s, 1.3s, 7.1s$ for buffers of $300, 500, 1K$, respectively, which can speed up to 188 by GPU [61].

4.2. Noise-Free Performance

Table 1 compares our SPR and Self-Replay's performance against Gdumb's reported performances on MNIST and CIFAR-10. Interestingly, our Self-Replay performs better than Gdumb, showing great promise in the direction of self-superved continual learning in general. However, SPR's performance is below that of Gdumb when completely noise free. We speculate SPR's mechanics to retain clean samples lead to a tradeoff with precise class feature coverage which seems to be of relative importance in a noise-free setting.

	MNIST	CIFAR-10
Gdumb [54]	**91.9**	45.8
Self-Replay	88.9	**47.4**
SPR	85.5	44.3

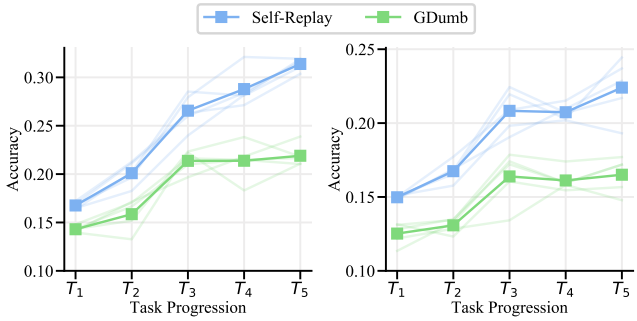
Table 1. **Noise Free performances** of Self-Replay and SPR compared with Gdumb [54]'s reported performances. Buffer size is fixed to 500.

4.3. Noise Robustness Comparison

Figure 1 contrasts the noise robustness of the strongest and closest baseline GDumb to Self-Replay under 40% and 60% noise levels while removing the Self-Centered filter from our method. Even still, Self-Replay is much more robust against high amounts of noisy labels at every task, validating that Self-Replay alone is able to mitigate the harmful effects of noise to a great extent.

4.4. Features from ImageNet Pretrained Model

We would like to clarify that our scenario and approach is much different and novel in that, the algorithm assumes an online stream of data and no ground-truth data is available to supervisedly train a noise detector. Not only that, the data we have to work is very small (e.g., $300, 500, 1000$) as the purpose is for a Delayed buffer to set aside small amounts from a stream of data for verification by our self-supervisedly trained Expert model. This was also motivated by the empirical evidence that using a supervised learning



(a) Noise rate 40% (b) Noise rate 60%

Figure 1. **Noise Robustness** of Self-Replay and GDumb on CIFAR-10. Both models use conventional reservoir sampling (*i.e.*, uniform random sampling from the input data stream) for the replay (purified) buffer; that is, no purification of the input data is performed. The vivid plots indicate the mean of five random seed experiments.

technique such as AUM [53], INCV [8], and using an ImageNet supervisedly pre-trained model for extracting the features led to worthless performances in the Table 2, 3.

	CIFAR-10 symmetric		
noise rate (%)	20	40	60
ImageNet pretrained	-9.0	-7.0	3.0
Self-supervised	75.5	70.5	54.3

Table 2. **Filtered noisy label percentages** in the purified buffer. We compare filtering performances from the self-supervisedly learned features with the ones from the ImageNet pretrained features. We set $E_{max} = 5$.

4.5. Analyses of Stochastic Ensemble Size (E_{max})

Figure 2 displays the performance of Stochastic Ensemble by increasing the ensemble sizes (E_{max}) from 1 to 40. Stochastic Ensemble performs better in all ensemble sizes than the non-stochastic BMM in terms of the percentages of filtered noisy labels on both MNIST and CIFAR10 with 60% noisy labels. A substantial boost is seen in the filtering performance up to 10. After 20, the performance starts to plateau on both MNIST and CIFAR-10. The empirically suggested optimal number of E_{max} may be around 20 for both MNIST and CIFAR-10 and this is further confirmed in Table 3 where we fix $E_{max} = 20$ and the overall filtering percentage increase by 2.4% on average, compared to the results in the main draft with $E_{max} = 5$.

4.6. Filtering performances on CIFAR-100.

Table 4 compares the filtering performances of SPR with the two state-of-the-art label filtering methods [53, 8] on

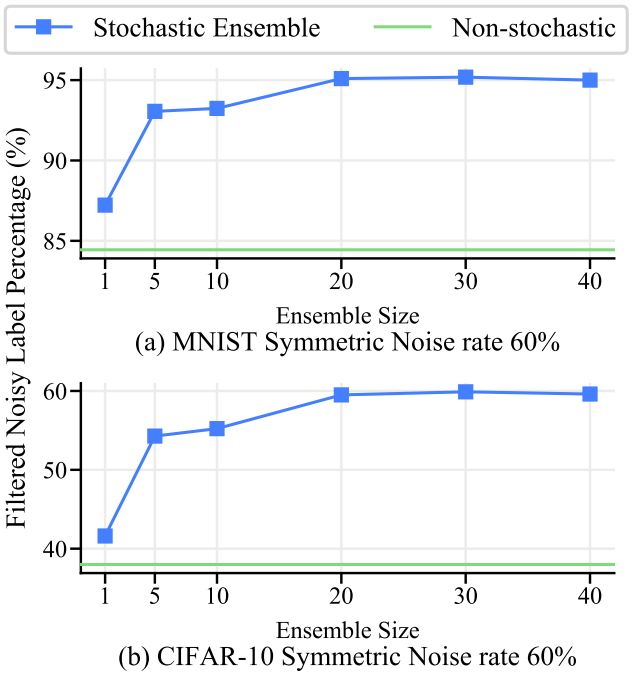
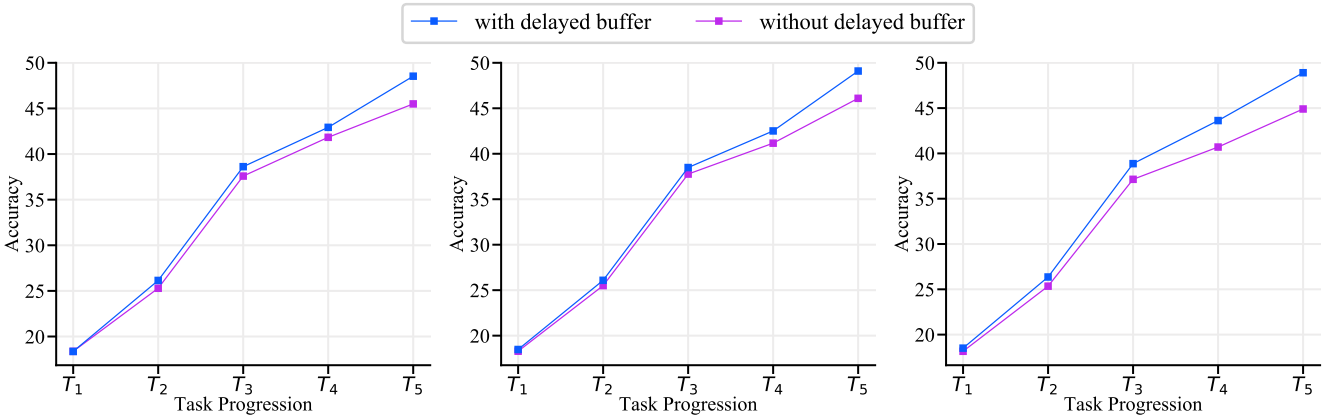


Figure 2. **Filtered noisy label percentages** in the purified buffer by increasing the ensemble size (E_{max}) on MNIST and CIFAR-10 with 60% noise rate. Stochastic Ensemble significantly performs better than the static version.

	MNIST					CIFAR-10				
	symmetric			asymmetric		symmetric			asymmetric	
noise rate (%)	20	40	60	20	40	20	40	60	20	40
AUM [53]	7.0	16.0	11.7	30.0	29.5	36.0	24.0	11.7	46.0	30.0
INCV [8]	23.0	22.5	14.3	37.0	31.5	22.0	18.5	9.3	37.0	30.0
Non-stochastic	79.5	96.3	84.5	96.0	88.5	50.5	54.5	38.0	53.0	50.5
SPR (Ours)	**95.0**	**96.8**	**95.0**	**99.9**	**97.5**	**79.5**	**76.3**	**59.5**	**72.0**	**59.0**

Table 3. **Filtered noisy label percentages** in the purified buffer. We compare SPR to two other state-of-the-art label filtering methods. We set $E_{max} = 20$.

CIFAR-100. SPR performs the best in all random symmetric noise and superclass symmetric noise with different levels of 20%, 40%, and 60%. Even the filtering performance on CIFAR-100 is superior to CIFAR-10. We believe this result is mainly due to the classes in CIFAR100 being more specific than CIFAR10 (e.g., automobile, airplane, bird in CIFAR10 where CIFAR100 has the trees superclass divided into maple, oak, palm, pine, willow), allowing SPR to self-supervisedly learn much more distinct features per class. This result is further reinforced on the WebVision dataset where SPR shows a weakness in filtering abstract classes such as "Spiral, in which the details can be found in Sec 4.8.



(a) Noise rate 20% (b) Noise rate 40% (c) Noise rate 60%

Figure 3. The overall accuracy of SPR over sequential task progression on CIFAR-10 with different noise rates. Training with the delay buffer means that self-supervised learning is performed using the samples in both the delay buffer and the purified buffer, whereas training without the delay buffer means it is done with the samples in the purified buffer only.

	random symmetric			superclass symmetric		
noise rate (%)	20	40	60	20	40	60
AUM [53]	33.5	46.8	13.9	25.0	21.4	32.4
INCV [8]	46.9	34.8	22.2	33.7	27.0	15.4
SPR	**82.9**	**79.6**	**64.8**	**76.5**	**69.4**	**56.0**

Table 4. **Filtered noisy label percentages** in the purified buffer. We compare SPR to two other state-of-the-art label filtering methods on CIFAR-100. We set $E_{max} = 5$. The buffer size is set to 5000. "random symmetric" refers to noise randomized across the 100 classes, while "superclass symmetric" refers to noise randomized within the CIFAR-100 superclasses [34, 36].

4.7. Self-Replay with Noisy Labeled Data

Table 5 compares the overall accuracy of Self-Replay when self-supervised training is performed with and without the delay buffer. Training with the delay buffer means using the samples in both the delay buffer B_d (red) and the purified buffer B_p (blue). In contrast, training without the delay buffer means using purified samples B_p (blue) only. We remind the normalized temperature-scaled cross-entropy loss in the main manuscript as

$$L_{self} = -\sum_{i=1}^{2(B_d+B_p)} \log \frac{e^{u_i^T u_j/\tau}}{\sum_{k=1}^{2(B_d+B_p)} \mathbb{1}_{k\neq i} e^{u_i^T u_k/\tau}}. \quad (6)$$

We observe an approximately 0.6% increase in MNIST and 3.3% increase in CIFAR-10 when using the delay buffer as well, even though it contains noisy labeled samples. We speculate that slight improvement is attained in MNIST due to the simplicity of the features. On the other hand, noticeable margins are seen in CIFAR-10, which we further analyze on a per-task basis, shown in Figure 3. The gaps are small in the earlier tasks but become more prominent as more tasks are seen. Moreover, the differences are even more significant when the level of noise rate increases. The take-home message here is that self-supervised training can benefit from the increased data even if it could possibly contain noisy labels.

	MNIST symmetric			CIFAR-10 symmetric		
noise rate (%)	20	40	60	20	40	60
SR with DB	91.0	91.8	91.1	48.5	49.1	48.9
SR without DB	90.3	91.0	90.5	45.5	46.1	44.9

Table 5. The overall accuracy of SPR with or without the samples in the delay buffer (DB). Self-supervised training can more benefit from more data even though some of them are possibly noisy

4.8. Analyses of the Results on WebVision

In the main manuscript, we briefly discuss the observation that Self-Replay and the Self-Centered filter do not synergize well on the WebVision dataset. In this section, we provide extended discussions about this behavior with qualitative and quantitative analyses.

Qualitative Analysis. We pointed out that classes such as "Spiral" or "Cinema" are highly abstract by overarching broad related knowledge, which is at the same time corrupted by noise. We show 50 random training data in Figure 5 and Figure 7 for "Spiral" and "Cinema", respectively. The Self-Centered filter samples for the same classes are also shown in Figure 6 and Figure 8. As visualized, it is not easy to interpret what the central concept is.

This is contrasted by the training samples in the classes

	GDumb	Self-Replay	Self-Centered filter	SPR
"Cinema"	34.3	46.4	19.6	26.8
"Spiral"	8.6	23.2	4.8	9.0
"ATM"	23.6	52.8	26.5	54.0
"Frog"	33.0	52.4	45.2	55.0

Table 6. Comparison of random sampling based methods (GDumb and Self-Replay) and the methods using the proposed Self-Centered filtering technique (Self-Centered filter and SPR). Random sampling is better for abstract classes such as "Cinema" and "Spiral", whereas Self-Centered filtering is better for ordinary noisy classes such as "ATM" or "Frog". The results are the mean of five unique random seed experiments.

"ATM" and "Frog" in Figure 11 and Figure 9. The classes contain noisy samples but represent the class concept without a high amount of abstraction. We also show the Self-Centered filter samples for the classes in Figure 12 and Figure 10. It is much more visually evident what class the samples represent.

Quantitative Analysis. Table 6 contrasts the performance of the two topics on GDumb, Self-Replay, Self-Centered filter, and SPR. The Self-Centered filter and SPR use the proposed Self-Centered filtering technique, whereas GDumb and Self-Replay use random sampling instead. The performances also support that random sampling may be a better performer for noisy and abstract classes, as GDumb and Self-Replay attain better performances. On the other hand, for ordinary noisy classes such as "ATM" or "Frog," the Self-Centered filter and SPR perform stronger than random sampling and show a synergetic effect.

4.9. Episode Robustness

Table 7 (episode B) and Table 8 (episode C) report the results of two different randomly permuted episodes. We include all of GDumb [54] combinations and the single best performing combination of PRS [30] and CRS [66] for each dataset. Even in two additional random episode experiments, SPR performs much stronger than all the baselines on all datasets with real, symmetric, or asymmetric noise.

4.10. Buffer Size Analysis

SPR requires a larger amount of memory than some baselines (excluding L2R), but the usage of the memory is different in that, a hold-out memory (Delay Buffer) is used for the purpose of filtering out the noisy labels, while only the Purified Buffer is used to mitigate the amount of forgetting. Hence, simply giving the other baselines a replay buffer twice as big would not be a fair comparison in the viewpoint of continual learning alone. Nonetheless, we run the experiments shown in Table 10, where all of GDumb [54] combinations are allowed twice the buffer size for replay. Even so, SPR using half the buffer size is able to outperform all the other baselines. Furthermore, to inform how the buffer size affects the results, we halve the original used buffer size and report the results in Table 9. SPR still strongly outperforms the baselines in all the datasets and noise rates. These two experiments show that SPR is robust to the buffer size, and its performance is due to self-supervised learning and the clean-buffer management, rather than using the hold-out memory for the Delay buffer.

4.11. Variance

Figure 4 visualizes the variances of top-3 best-performing methods for MNIST, CIFAR-10 with 40% symmetric noise rate, and WebVision with real-noise. Among the symmetric noise experiments with five different random seeds, SPR shows a minor amount of variance throughout the tasks. However, for WebVision, a noticeable amount of fluctuations are seen for all three approaches.

	MNIST					CIFAR-10					WebVision
	symmetric			asymmetric		symmetric			asymmetric		real noise
noise rate (%)	20	40	60	20	40	20	40	60	20	40	unknown
Multitask 0% noise [5]			98.6					84.7			-
Multitask [5]	94.5	90.5	79.8	93.4	81.1	65.6	46.7	30.0	77.0	68.7	55.5
CRS + L2R [57]	80.8	74.1	59.7	85.3	79.8	29.8	23.1	16.0	36.4	36.1	-
CRS + Pencil [72]	-	-	-	-	-	-	-	-	-	-	25.1
PRS + L2R [57]	80.7	74.0	60.4	83.2	80.1	30.8	22.8	15.0	36.3	32.9	-
PRS + Pencil [72]	-	-	-	-	-	-	-	-	-	-	26.5
MIR + L2R [57]	79.6	68.6	51.6	83.2	79.5	31.1	21.0	14.5	34.7	33.6	-
MIR + Pencil [72]	-	-	-	-	-	-	-	-	-	-	22.6
GDumb [54]	70.1	54.6	32.3	78.2	71.1	29.6	22.4	16.5	33.0	30.9	33.3
GDumb + L2R [57]	67.1	59.2	40.6	70.6	68.7	27.0	25.5	21.8	29.9	29.4	-
GDumb + Pencil [72]	70.2	53.9	35.4	77.5	70.2	28.1	21.0	15.9	31.5	30.6	27.5
GDumb + SL [68]	65.6	47.5	30.5	73.3	68.5	27.1	22.6	16.8	33.2	31.4	32.5
GDumb + JoCoR [69]	68.3	56.0	41.0	78.5	70.9	26.6	21.1	15.9	32.9	32.2	22.9
SPR	**86.8**	**87.2**	**82.1**	**86.6**	**85.5**	**42.0**	**42.4**	**39.1**	**44.4**	**43.3**	**41.6**

Table 7. **Overall accuracy on episode B** after all sequences of tasks are trained. The buffer size is set to 300, 500, 1000 for MNIST, CIFAR-10, and WebVision, respectively. We report all of GDumb [54] combinations and single best performing combination of PRS [30] and CRS [66]. Some empty slots on WebVision are due to the unavailability of clean samples required by L2R for training [57]. The results are the mean of five unique random seed experiments.

	MNIST					CIFAR-10					WebVision
	symmetric			asymmetric		symmetric			asymmetric		real noise
noise rate (%)	20	40	60	20	40	20	40	60	20	40	unknown
Multitask 0% noise [5]			98.6					84.7			-
Multitask [5]	94.5	90.5	79.8	93.4	81.1	65.6	46.7	30.0	77.0	68.7	55.5
CRS + L2R [57]	79.9	74.9	58.2	84.4	79.4	29.3	24.4	16.8	37.2	37.5	-
CRS + Pencil [72]	-	-	-	-	-	-	-	-	-	-	29.9
PRS + L2R [57]	80.5	72.3	55.2	83.8	80.1	30.6	23.3	16.3	37.2	36.1	-
PRS + Pencil [72]	-	-	-	-	-	-	-	-	-	-	28.5
MIR + L2R [57]	80.3	69.7	47.1	83.0	77.6	28.2	21.3	15.6	36.3	34.3	-
MIR + Pencil [72]	-	-	-	-	-	-	-	-	-	-	22.4
GDumb [54]	71.8	52.8	37.5	79.2	72.1	28.7	23.0	16.3	34.2	31.9	31.6
GDumb + L2R [57]	67.7	58.2	42.7	69.3	67.6	28.9	24.8	19.7	31.8	29.4	-
GDumb + Pencil [72]	69.0	54.2	37.8	78.6	71.2	27.5	21.0	16.6	31.3	31.8	28.5
GDumb + SL [68]	65.4	48.4	29.1	72.4	67.7	28.3	22.9	15.0	31.4	31.9	31.6
GDumb + JoCoR [69]	70.4	59.0	40.6	77.4	70.6	27.8	22.3	15.5	33.4	31.7	24.3
SPR	**86.6**	**87.5**	**84.4**	**87.0**	**87.3**	**43.7**	**43.1**	**39.8**	**44.3**	**43.2**	**40.2**

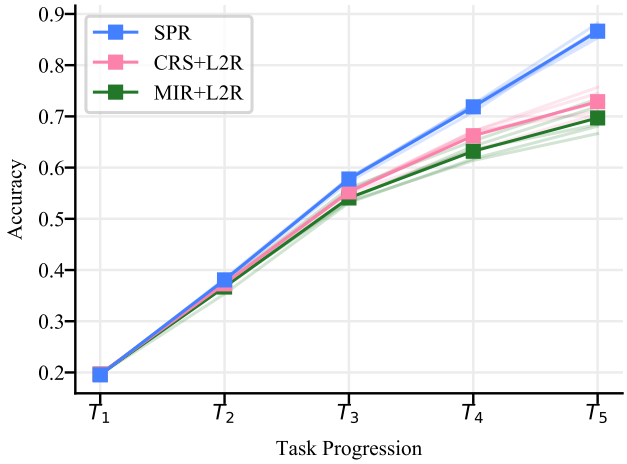
Table 8. **Overall accuracy on episode C** after all sequences of tasks are trained. The buffer size is set to 300, 500, 1000 for MNIST, CIFAR-10, and WebVision, respectively. We report all of GDumb [54] combinations and single best performing combination of PRS [30] and CRS [66]. Some empty slots on WebVision are due to the unavailability of clean samples required by L2R for training [57]. The results are the mean of five unique random seed experiments.

	MNIST					CIFAR-10					WebVision
	symmetric			asymmetric		symmetric			asymmetric		real noise
Buffer size	150			150		250			250		500
noise rate (%)	20	40	60	20	40	20	40	60	20	40	unknown
GDumb + L2R [57]	64.8	55.5	37.8	71.2	66.8	23.2	22.1	19.3	28.4	24.8	-
GDumb + Pencil [72]	59.3	48.1	36.4	76.4	66.6	25.6	17.9	13.9	27.6	26.8	21.1
GDumb + SL [68]	61.5	41.3	31.1	66.8	56.8	20.7	19.8	18.8	29.2	26.4	26.4
GDumb + JoCoR [69]	66.8	60.9	33.0	74.4	66.3	23.8	18.9	14.2	26.2	26.2	23.0
SPR	**82.6**	**85.4**	**81.2**	**77.0**	**81.6**	**41.2**	**41.2**	**37.8**	**42.8**	**41.3**	**39.4**

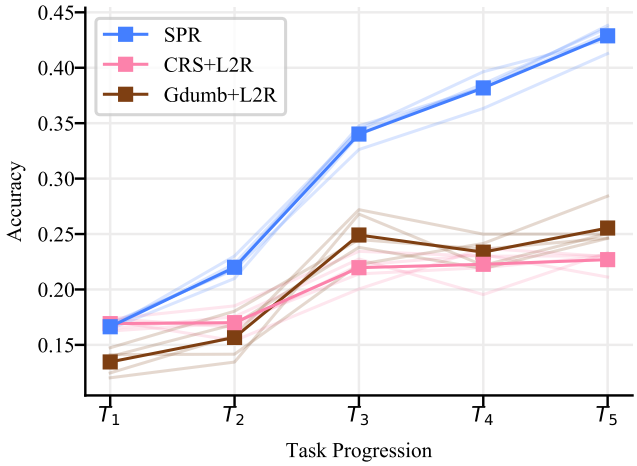
Table 9. **Overall accuracy on the half buffer size** after all sequences of tasks are trained. The buffer size is set to 150, 250, 500 for MNIST, CIFAR-10, and WebVision, respectively. We report all of GDumb [54] combinations. An empty slot on WebVision are due to the unavailability of clean samples required by L2R for training [57].

	MNIST					CIFAR-10					WebVision
	symmetric			asymmetric		symmetric			asymmetric		real noise
Buffer size	600			600		1000			1000		2000
noise rate (%)	20	40	60	20	40	20	40	60	20	40	unknown
GDumb + L2R [57]	76.7	62.6	51.9	79.7	73.3	31.4	27.3	24.0	35.0	36.0	-
GDumb + Pencil [72]	72.1	58.5	39.4	75.3	73.5	31.2	24.5	16.4	38.6	35.5	33.0
GDumb + SL [68]	66.0	47.2	31.7	79.0	74.8	33.1	23.2	17.7	40.4	37.3	38.5
GDumb + JoCoR [69]	74.3	57.8	42.5	78.3	76.0	31.9	22.8	17.4	42.5	38.1	27.0
Buffer size	300			300		500			500		1000
SPR	**85.4**	**86.7**	**84.8**	**86.8**	**86.0**	**43.9**	**43.0**	**40.0**	**44.5**	**43.9**	**40.0**

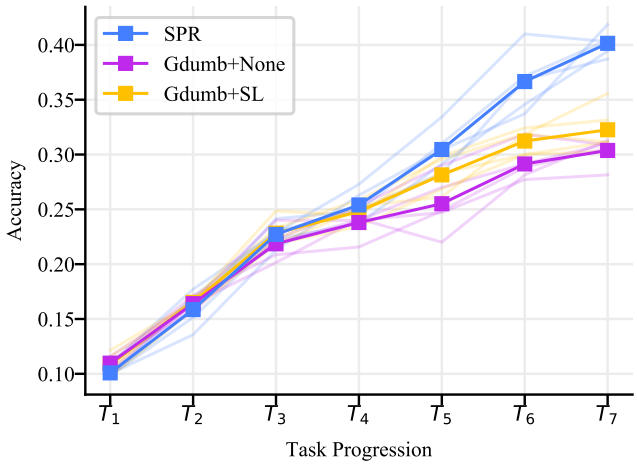
Table 10. **Overall accuracy on the double buffer size for all of GDumb combinations** after all sequences of tasks are trained. The buffer size is set to 600, 1000, 2000 for MNIST, CIFAR-10, and WebVision, respectively. An empty slot on WebVision are due to the unavailability of clean samples required by L2R for training [57]. Note that SPR outperforms all of GDumb [54] combinations with the buffer size of 300, 500, 1000 for MNIST, CIFAR-10, and WebVision, respectively.



(a) MNIST Symmetric Noise rate 40%



(b) CIFAR-10 Symmetric Noise rate 40%



(c) WebVision

Figure 4. Accuracy and variances of top-3 best-performing methods for MNIST, CIFAR-10 and WebVision.

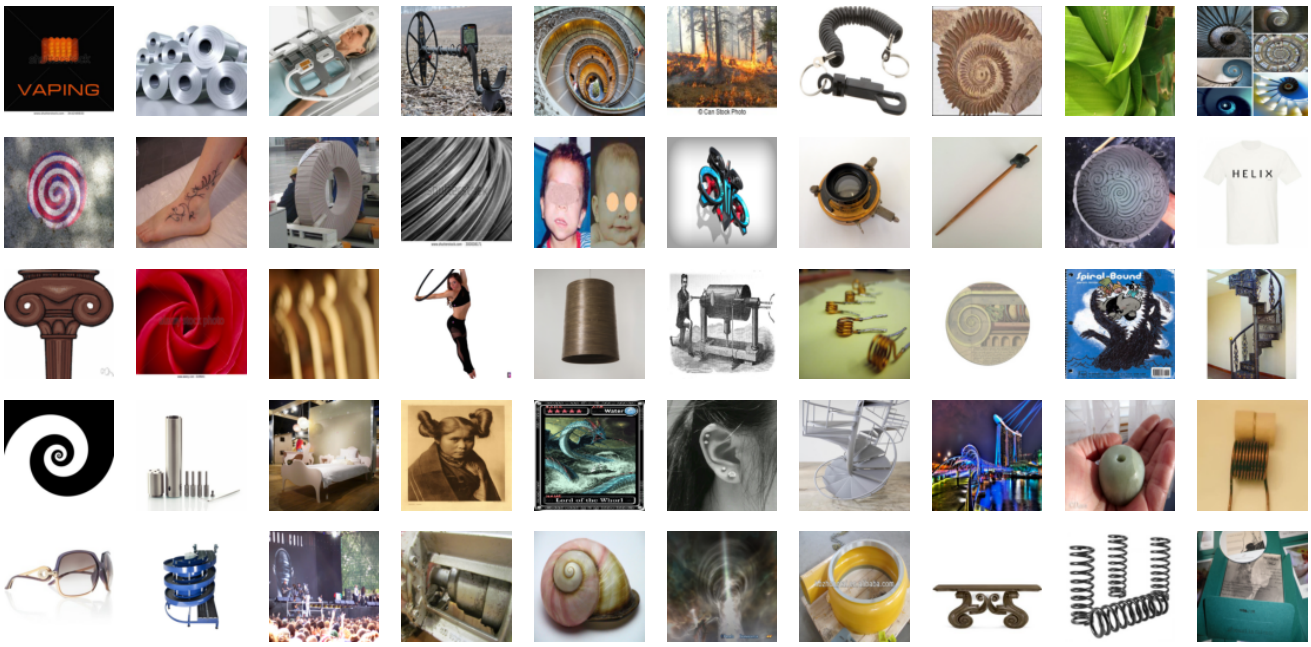


Figure 5. 50 random samples of the “Spiral” class from the training set.

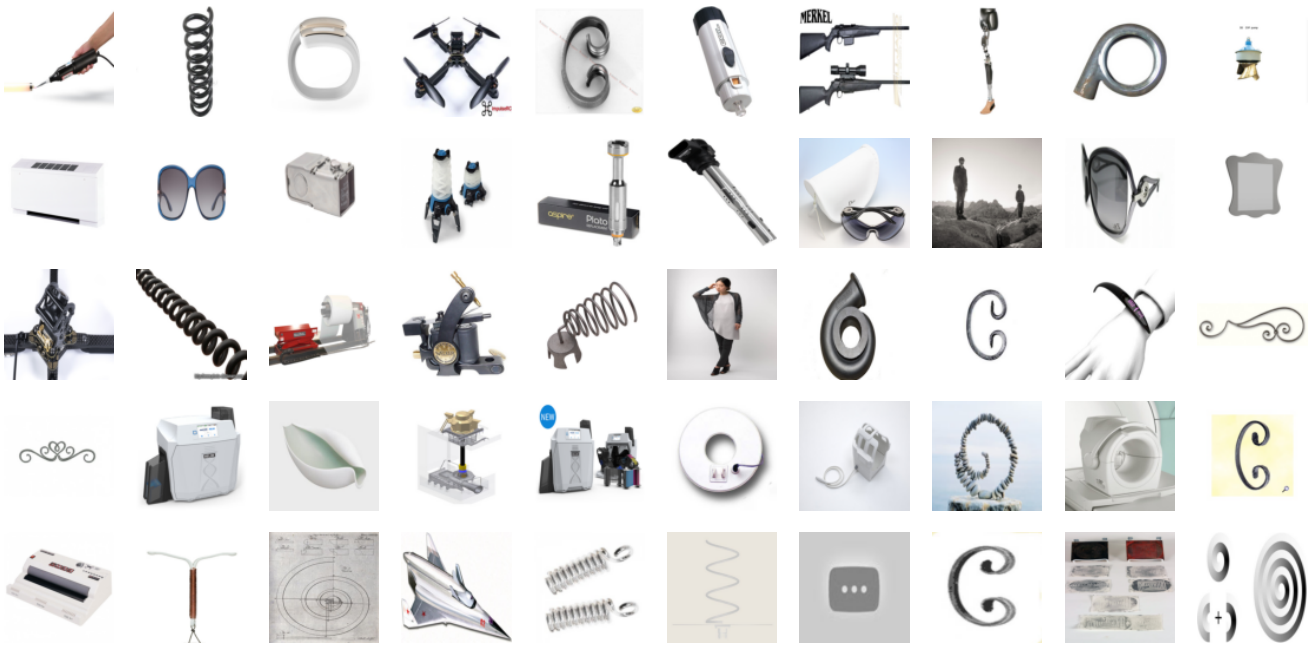


Figure 6. 50 random training samples of the “Spiral” class from the purified buffer.

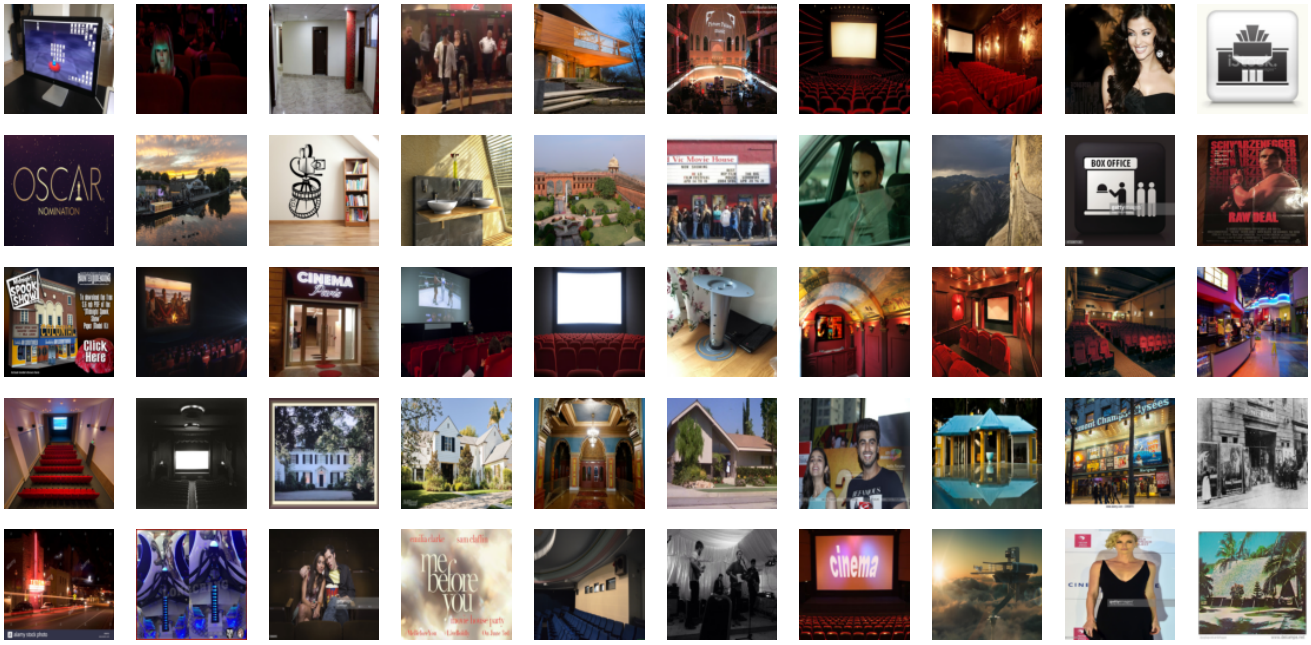


Figure 7. 50 random samples of the “Cinema” class from the training set.

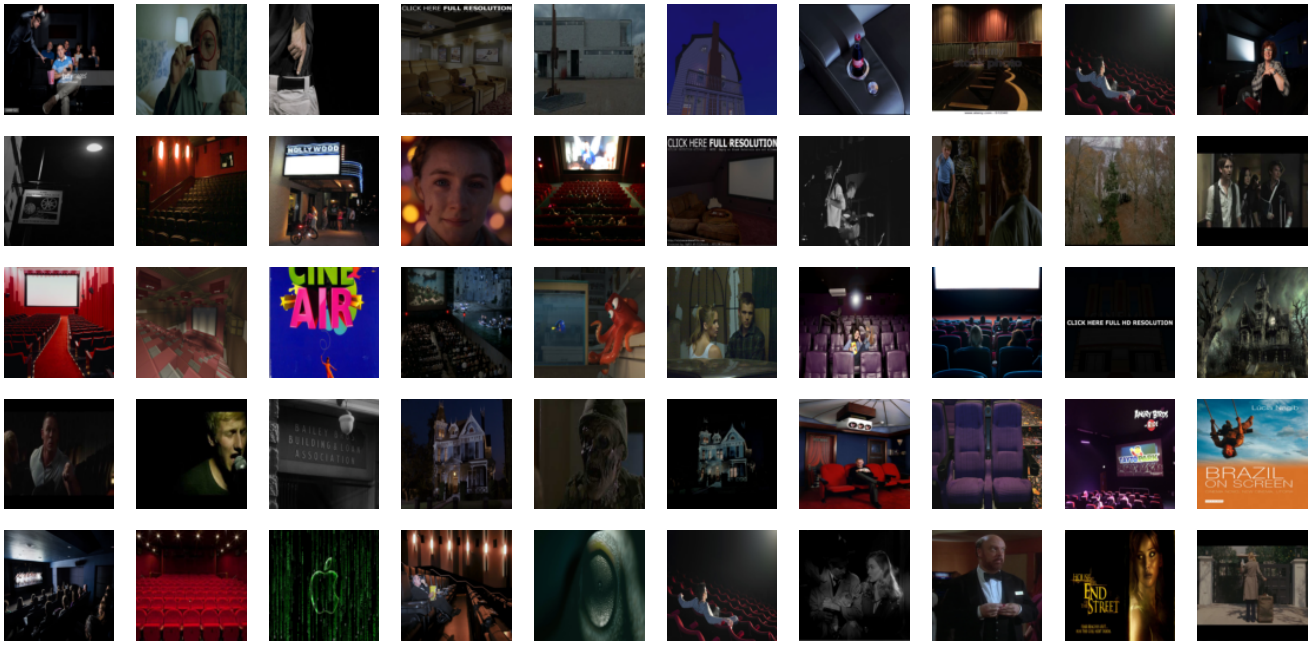


Figure 8. 50 random training samples of the “Cinema” class from the purified buffer.

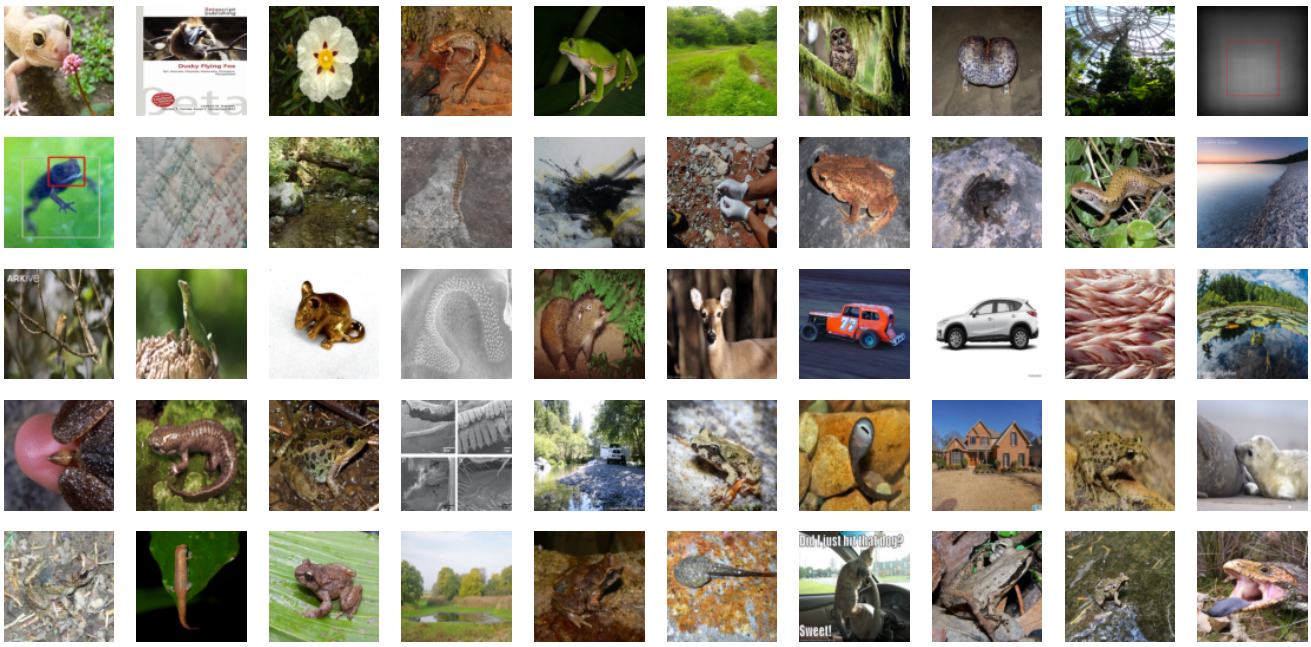


Figure 9. 50 samples of the “Frog” class from the training set.

Figure 10. 50 training samples of the “Frog” class from the purified buffer.

Figure 11. 50 samples of the “ATM” class from the training set

Figure 12. 50 random training samples of the “ATM” class from the purified buffer.

References

[1] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars. Online continual learning with maximally interfered retrieval. In *NeurIPS*, 2019. 2

[2] R. Aljundi, R. Marcus, and T. Tuytelaars. Selfless sequential learning. In *ICLR*, 2019. 1

[3] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. Unsupervised label noise modeling and loss correction. In *ICML*, 2019. 2

[4] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *ICML*, 2017. 2

[5] R. Caruaca. Multitask learning. *Machine Learning*, 28:41–75, 1997. 3, 7

[6] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 2

[7] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486v4*, 2019. 2

[8] P. Chen, B. Liao, G. Chen, and S. Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*, 2019. 2, 3, 4, 5

[9] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2, 3

[10] T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, 2019. 2

[11] C. d'Autume, S. Ruder, L. Kong, and D. Yogatama. Episodic memory in lifelong language learning. In *NeurIPS*, 2019. 2

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1:1–38, 1991. 1

[13] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2016. 2

[14] Enrico Fini, Stphane Lathuilire, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extremem memory constraints. In *ECCV*, 2020. 1

[15] Y. Ge, D. Chen, and H. Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. In *ICLR*, 2020. 2

[16] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2

[17] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017. 2

[18] G. Gupta, K. Yadav, and L. Paull. La-maml: Look-ahead meta learning for continual learning. In *NeurIPS*, 2020. 2

[19] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 2018. 2

[20] J. Han, P. Luo, and X. Wang. Deep self-learning from noisy labels. In *ICCV*, 2019. 2

[21] H. Harutyunyan, K. Reing, G. V. Steeg, and A. Galstyan. Improving generalization by controlling label-noise information in neural network weights. In *ICML*, 2020. 2

[22] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019. 2

[23] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2

[24] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NIPS*, 2018. 2

[25] J. Huang, L. Qu, R. Jia, and B. Zhao. O2u-net: A simple noisy label detection approach for deep neural networks. In *ICCV*, 2019. 2

[26] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Comput*, 3:79–87, 1991. 1

[27] K. Javed and M. White. Meta-learning representations for continual learning. In *NeurIPS*, 2019. 2

[28] L. Jiang, D. Huang, M. Liu, and W. Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *ICML*, 2020. 2

[29] L. jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei. Mentornet:learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018. 2

[30] D. Kim, J. Jeong, and G. Kim. Imbalanced continual learning with partioning reservoir sampling. In *ECCV*, 2020. 2, 3, 6, 7

[31] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 3

[32] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the National Academy of Sciences*, 2017. 1

[33] J. Kremer, F. Sha, and C. Igel. Robust active label correction. In *AISTATS*, 2018. 2

[34] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto, 2009. 2, 5

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. In *IEEE*, 1998. 2

[36] S. Lee, J. Ha, D. Zhang, and G. Kim. A neural dirichlet process mixture model for task-free continual learning. In *ICLR*, 2020. 1, 5

[37] J. Li, Y. Wong, Q. Zhao, and M. Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, 2019. 2

[38] J. Li, C. Xiong, and S. Hoi. Mopro: Webly supervised learning with momentum prototypes. In *ICLR*, 2021. 2

[39] J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. H. Hoi. Prototypical contrastive learning of unsupervised representations. In *ICLR*, 2020. 2

[40] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv: 1708.02862*, 2017. 2

[41] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L. Li. Learning from noisy labels with distillation. In *ICCV*, 2017. 2

[42] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016. 1

[43] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017. 2

[44] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 3

[45] M. Lukasik, S. Bhojanapalli, A. K. Menon, and S. Kumar. Does label smoothing mitigate label noise? In *ICML*, 2020. 2

[46] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S. Xia, S. Wijewickrema, and J. Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, 2018. 2

[47] E. Malach and S. Shalev-Shwartz. Decoupling "when to update" from "how to update". In *NeurIPS*, 2017. 2

[48] D. Mandal, S. Bharadwaj, and S. Biswas. A novel self-supervised re-labeling approach for training with noisy labels. In *WACV*, 2020. 2

[49] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox. Self: Learning to filter noisy labels with self-ensembling. In *ICLR*, 2019. 2

[50] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2017. 2

[51] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[52] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: a loss correction approach. In *CVPR*, 2017. 2

[53] G. Pleiss, T. Zhang, E. R. Elenberg, and K. Q. Weinberger. Identifying mislabeled data using the area under the margin ranking. In *NIPS*, 2020. 2, 3, 4, 5

[54] A. Prabhu, P. H.S. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2019. 2, 3, 6, 7, 8

[55] Aljundi Rahaf, Min Lin, Baptiste Goujaud, and Bengio Yoshua. Gradient based sample selection for online continual learning. In *NeurIPS*, 2019. 2

[56] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR workshop*, 2015. 2

[57] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 2, 3, 7, 8

[58] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*, 2019. 2

[59] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 1

[60] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *NeurIPS*, 2017. 2

[61] Gustavo Rodrigues Lacerda Silva, Rafael Ribeiro De Medeiros, Brayan Rene Acevedo Jaimes, Carla Caldeira Takahashi, Douglas Alexandre Gomes Vieira, and AntôNio De PáDua Braga. Cuda-based parallelization of power iteration clustering for large datasets. *IEEE Access*, 5:27263–27271, 2017. 3

[62] H. Song, M. Kim, and J. Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *ICML*, 2019. 2

[63] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018. 2

[64] B. Tang and D. S. Matteson. Graph-based continual learning. In *ICLR*, 2021. 2

[65] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, 2017. 2

[66] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. 6, 7

[67] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S. Xia. Iterative learning with open-set noisy labels. In *CVPR*, 2018. 2

[68] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. Symmetric cross entropy for robust learning with noisy labels. In *ICCV*, 2019. 3, 7, 8

[69] H. Wei, L. Feng, X. Chen, and B. An. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, 2020. 2, 3, 7, 8

[70] Y. Xu, P. Cao, Y. Kong, and Y. Wang. L_dmi: An information-theoretic noise-robust loss function. In *NeurIPS*, 2019. 2

[71] M. Ye, X. Zhang, P. C. Yuen, and S. Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, 2019. 2

[72] K. Yi and J. Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR*, 2019. 2, 3, 7, 8

[73] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018. 1

[74] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama. How does disagreement help generalization against label corruption? In *ICML*, 2019. 2

[75] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classiers with localizable features. In *ICCV*, 2019. 3

[76] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 2

[77] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NIPS*, 2018. 2