# May the Forgetting Be with You: Alternate Replay for Learning with Noisy Labels – Supplementary materials

Monica Millunzi[1,2]
monica.millunzi@unimore.it
Lorenzo Bonicelli[1]
lorenzo.bonicelli@unimore.it
Angelo Porrello[1]
angelo.porrello@unimore.it
Jacopo Credi[2]
jacopo.credi@axyon.ai
Petter N. Kolm[3]
petter.kolm@nyu.edu
Simone Calderara[1]
simone.calderara@unimore.it

[1] AImageLab
University of Modena and Reggio Emilia
Modena, Italy
[2] AxyonAI
Viale dell'Autodromo, 210
Modena, Italy
[3] Courant Institute of Mathematical Science
New York University (NYU)
New York, USA

## A On the effectiveness of buffer consolidation

By combining AER with ABS we obtain a balance between purity – for samples of the current task – while preserving the complexity of those from the past. To achieve this, the backbone network had to be trained on a stream of noisy data. While we find that the effect of noise from the current task is mitigated by AER (Appendix C), we can further reduce its influence with the help of the memory buffer.

In principle, with an ideal sample selection strategy we could simply train on samples from $\mathcal{M}$ to adjust the predictions of the network at the end of the task in a fully-supervised fashion (**buffer fit.**). While we empirically find in Sec. 4 that such a strategy delivers remarkable results, we can refine it to handle more complex noise scenarios.

In particular, we use a modified version of MixMatch [3] to obtain a more robust model, using the most *uncertain* samples as a source for unlabeled data. Similarly to [1], we fit a two-component Gaussian Mixture Model (GMM) $g(\mathcal{L})$ on the loss $\mathcal{L}$ of each $(\mathbf{x}, \tilde{y}) \in \mathcal{M}$. Then, we compute the perceived uncertainty of each sample $u(\mathbf{x})$ as the posterior $g(l|\mathcal{L})$, where $l$ indicates the Gaussian component with the smaller mean. Samples are then separated into *pure* $\mathcal{P}$ and *uncertain* $\mathcal{U}$ with a simple threshold on $g(l|\mathcal{L})$.

From this, samples in $\mathcal{P}$ have label $\tilde{y} \approx y$, thus we can use them to compute a supervised loss term. Instead, for $\mathbf{x} \in \mathcal{U}$ we compute $\hat{y}$ using the model's response on different

| CIFAR-100 | **Parameter $\alpha$** | | | | | |
|---|---|---|---|---|---|---|
| | **0%** | **25%** | **50%** | **60%** | **75%** | **90%** |
| *Asym 40%* | 24.76± 0.14 | 26.69± 0.26 | 28.76± 0.51 | 29.70± 0.61 | 29.26± 0.91 | 29.90± 0.53 |
| *Sym 40%* | 27.01± 0.84 | 31.99± 0.62 | 36.92± 0.55 | 38.52± 0.70 | 38.64± 0.57 | 39.40± 0.70 |
| *Sym 60%* | 15.30± 0.44 | 17.35± 0.06 | 20.74± 0.48 | 24.61± 0.55 | 26.34± 0.85 | 30.67± 1.26 |

Table A: FAA [↑] on CIFAR100 with varying noise to assess the influence of $\alpha$

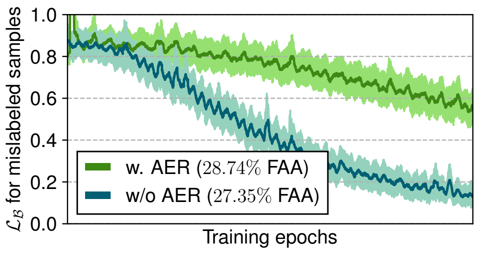

Figure A: Effect of AER on the speed at which the model learns the noisy data

augmentations $T$ of $\mathbf{x}$:

$$\hat{y} = u(\mathbf{x})\tilde{y} + \frac{1-u(\mathbf{x})}{\eta}\sum_{i=1}^{\eta} f_\theta(T(\mathbf{x})), \tag{1}$$

Finally, we obtain the refined set $\mathcal{Q} = \{(\mathbf{x},\hat{y}) : (\mathbf{x},\tilde{y}) \in \mathcal{U}\}$ and follow up with the MixMatch procedure to compute the supervised and self-supervised loss terms $\mathcal{L}_s$ and $\mathcal{L}_u$ respectively. The overall loss term is computed as $\mathcal{L}_s + \lambda_u\mathcal{L}_u$, where $\lambda_u$ is a regularization hyperparameter.

## B On the influence of the hyperparameter $\alpha$

In this section, we want to carry out a sensitivity analysis targeting the value of $\alpha$. Recall that alpha controls the proportion of samples to be discarded from the insertion phase within the buffer. We here report the results yielded by several $\alpha$ values under three different noise settings (asymm. 40%, symm. 40%, symm. 60%). The experiments, reported in Tab. A, are conducted on Split CIFAR-100, with performance measured in terms of Final Average Accuracy. It can be concluded that $\alpha >= 50\%$ is a good choice, with gains that stabilize around $60\% - 70\%$. In our experiments, we remark that we avoided tuning $\alpha$ and set the same value for every dataset/noise ratio/noise type.

## C On the effectiveness of AER as a regularizer for CNL

Here, we further provide evidence of the impact of AER on the overall performance of the model. In particular, in Fig. A we depict the final accuracy (FAA) and the loss of the noisy samples from the current task of ER-ACE with and without AER during the second task of Split CIFAR-10.

| **Computational Cost** | **PuriDivER** | **PuriDivER.ME** | **OURs** |
|---|---|---|---|
| Total Time (*hours*) | 5h15m | 2h50m | **1h30m** |
| Epoch Time (*seconds*) | 76.90s | **15.69s** | 18.56s |
| Task Time (*minutes*) | 73m50s | 19m39s | **16m33s** |
| Memory Used (*GB*) | 7.77 | 7.50 | **6.75** |

Table B: Comparison of Computational Cost [↓] of different methods when trained on CIFAR-10 with 40% noise.

Surprisingly, we find that AER vastly reduces the rate of convergence of noisy samples, which just by itself improves over the baseline in terms of FAA. Indeed, in rehearsal CNL providing a purified and diverse set of examples to counter forgetting is only part of the challenge: as the model is subjected to a continuous stream of noisy data from the current task, an important effect is to reduce the speed with which noisy samples from the present are learned.

## D On the computational demands of CLN methods

We perform all the experiments on a Tesla V100-SXM2-16GB GPU. In Tab. B we report the computational costs of different methods in our setting, in terms of runtime and consumed memory.

Not only our method achieves superior performance, as shown in Tab. 1, but also exhibits a lower overall training time.

## E Additional details on SPR and CNLL

In the main paper we provide a comparison between our proposal and SPR and CNLL. Nevertheless, as these methods were originally designed for the single-epoch setting, we had to design specific adjustments to make them viable for our scenario.

SPR initially stores samples in a *delayed buffer* – then splits into clean and noisy sets, with the former stored in a separate long-term buffer – and then optimizes the model for approximately 7,000 training iterations through a self-supervised (SSL) objective. This implies that SPR involves approximately 448$\times$ iterations than standard training[1], making it unfeasible for our scenario due to time constraints. Indeed, while on CIFAR-10 our method takes around 16 minutes to complete 1 task (Tab. B), SPR would require over 119 hours. We thus opt to distribute the training iterations of SPR across 25 epochs (see Tab. 2). Finally, as SPR employs two distinct memory buffers, we set the buffer size to 1000 for a fair comparison.

CNLL uses variable-length buffers to store confident clean and noisy samples, which implies a CL setting with unrestricted memory across tasks. To ensure fairness in comparison, we adhere to the well-established memory-budgeted CL [5, 7, 12] setting. Thus, for CNLL we allocate a total memory budget of 2,500 exemplars across all 5 buffers specified by the original method.

As we move from the single to multi-epoch setting, we find a reduced effectiveness of the regularization of CNLL; such a result is in line with our hypothesis of Sec. 3.1: as

[1]assuming a buffer size of 500, batch size of 32, and 10,000 samples

more epochs are allowed to learn the current task, sample selection based on the small-loss criterion fails to distinguish clean and noisy samples. Moreover, we find that such an outcome is maintained even in an unrestricted setting, where the memory budget is not a concern.

Finally, the performance gap w.r.t. our proposal is even more pronounced for SPR$^{\ddagger}$, where our method attains significantly higher accuracy in considerably less time; indeed, our reduced version of SPR requires around 109$\times$ more time than our proposal, in line with our estimation.

## F Additional details on the experimental settings and Final Forgetting

To evaluate our proposal we build upon the open-source codebase provided by Mammoth [4, 5, 6], a CL framework based on `PyTorch`.

### On the choice of datasets and noise

We empirically validate our method on four different classification benchmarks as mentioned in the main paper. For experiments on CIFAR-10/100 [10] and NTU-60 [11], we corrupt the labels of the datasets at hand to obtain different noise configurations, which we then keep fixed for each of the experiments for fairness of results comparison across multiple methods.

In the process of injecting symmetric noise, we replace the ground-truth label with probability $r \in [0,1]$ determined by the designated noise rate. The asymmetric or *class-dependent* noise setting is an approximation of real-world corruption patterns, which alters labels within the same superclass. For example, in the CIFAR-100 dataset, each image comes with a "fine" label (specific class) and a "coarse" label (superclass). Here, label transitions are parameterized by $r$ such that the wrong class and true class have probability $r$ and $1-r$, respectively. This results in sample ambiguity occurring only between similar classes, as it would in a realistic scenario.

In each experiment, samples from the main dataset are split into disjoint sets based on their class and organized into tasks, following the ClassIL setup. We obtain the following versions of the datasets.

***Seq. CIFAR-10*** The original dataset contains $50,000$ train and $10,000$ test low-resolution color images in 10 different classes. During training the model encounters 2 classes per task, namely (“airplane”, “car”), (“bird”, “cat”), (“deer”, “dog”), (“frog”, “horse”), (“ship”, “truck”).

***Seq. CIFAR-100*** This original dataset is like the CIFAR-10, except it has 100 classes with 600 images each. Images are grouped into 20 superclasses, thus each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). Following this categorization, we organize classes in 10 tasks, each containing 5 classes from the same superclass.

***Seq. NTU-60*** It comprises 60 action classes with 56,880 video samples, including 3D skeletal data (25 body joints per frame), all captured simultaneously using three Kinect V2 cameras. We here split the dataset into 6 tasks of 10 classes each.

***Seq. Food-101N*** The dataset is composed of 101 web-crawled food images, split into 5 tasks (the first 4 containing 20 classes and the latter containing the remaining 21). Each class is relatively balanced, with an average of around 523 images per class and a standard deviation of

| **Benchmark** | **Seq. CIFAR-100** | | | | | **Seq. NTU-60** | |
|---|---|---|---|---|---|---|---|
| | *symm* | | | *asymm* | | *symm* | |
| **Noise rate** | 20 | 40 | 60 | 20 | 40 | 20 | 40 |
| Joint | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Finetune | 81.52 | 71.51 | 57.96 | 73.91 | 54.71 | 85.09 | 73.46 |
| Reservoir | 55.88 | 55.79 | 44.90 | 43.48 | 35.22 | 54.53 | 61.33 |
| + *CoTeaching* | 55.20 | 54.95 | 36.07 | 54.77 | 26.03 | 34.30 | 29.03 |
| + *DivideMix* | 22.33 | 26.73 | 20.94 | 23.45 | 16.73 | 18.45 | 18.70 |
| PuriDivER | 20.52 | 18.21 | 14.77 | 22.51 | 17.26 | 41.29 | 34.25 |
| PuriDivER.ME$^\dagger$ | 24.34 | 25.06 | 26.83 | 25.40 | 21.82 | 25.76 | 18.41 |
| **OURs** | 22.89 | 21.26 | 22.13 | 21.19 | 16.90 | 12.94 | 14.05 |
| *w. consolidation* | <u>19.03</u> | <u>11.67</u> | <u>12.02</u> | <u>20.15</u> | <u>9.28</u> | <u>8.54</u> | <u>0.29</u> |

Table C: Final Forgetting (FF) [$\downarrow$] of CNL methods on our selection of bencharks. $^\dagger$ Additional baselines created by adapting existing loss-based and CL approaches to the multi-epoch scenario.

around 11 (totaling 52867 images resized to $224 \times 224$). The dataset contains instance-level noise, thus simulating a real-world scenario.

Notice that since some labels are incorrect, real class distribution for each task might vary. Details on the noisy labels injected on Seq. CIFAR-10/100, Seq. NTU-60 are released with the code.

## Training details

***Architecture*** We use ResNet [9] family as a backbone for all the methods involved in our evaluation. ResNet18 is used for CIFAR-10/100 and ResNet34 is used for Food-101N, as in [2]. All the experiments do not feature pretraining.
***Augmentation*** We apply random crops and horizontal flips to both stream and buffer examples, for each dataset at hand. For the implementations of PuriDivER, we use AutoAugment [8] as in the original paper [2].
***Training*** We deliberately hold batch size out of the hyperparameter space and keep it fixed to 32 for both stream and buffer examples. For each task, we train for 50 epochs for CIFAR-10/100, and 20 for Food-101N.
***Buffer consolidation with MixMatch*** At the end of each task, we finetune the model on the buffer examples only, for 255 epochs. During this stage, we use SGD with Warm Restart (SGDR) through Cosine Annealing and a batch size of 64. For the purpose of label co-refinement, we set the number of different augmentations $\eta$ of Eq. (1) to perform on the samples in the *uncertain* set to 3.

## Results in terms of Final Forgetting

We repeat each of the experiments five times. We report in Tab. 1 of the main paper the Final Average Accuracy for all the experiments, with standard error values.

We also provide the final forgetting measure in Eq. (2) for all methods of the main comparison in Tab. C.

$$FF \triangleq \frac{1}{T-1}\sum_{j=0}^{T-2} f_j, \text{s.t.} f_j = \max_{t\in 0,\dots,T-2} a_j^t - a_j^{T-1} \tag{2}$$

where $a_j^t$ is the accuracy of the model on the $j^{th}$ task after training on $t$ tasks. These additional results depict a **lower** degree of **forgetting** of our proposal w.r.t. the baselines.

When paired with Tab. 1 of the manuscript, such evidence shows higher overall effectiveness in learning from a noisy source of data, allowing more stable convergence on the current task and lower losses due to forgetting.

## G Hyperparameters

We choose to use different buffer sizes relying on the dataset length. For experiments conducted on CIFAR-10 and CIFAR-100, the buffer size is set to 500 and 2000, respectively. We set the buffer size to 500 for experiments on NTU. Finally, we use a buffer size of 2000 for Food-101.

We select the other hyperparameters by performing a grid search and using the Final Average Accuracy (FAA) as the selection criterion for the best parameters. Here, we report the best values for each model, categorized by dataset and noise type.

### CIFAR-10

Noise type: *sym* – 20%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.1; $\lambda_u$: 0.01
- **ER**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.1
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.01
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.03

Noise type: *sym* – 40%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.1; $\lambda_u$: 0.01

- **ER**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.03
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.03
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.03

Noise type: *sym* – 60%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.1; $\lambda_u$: 0.01
- **ER**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.1
- **ER + CoTeaching**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.03
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.03
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.01

## CIFAR-100

Noise type: *sym* – 20%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.05; $\lambda_u$: 0.01
- **DividERMix**: $lr$: 0.03
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.01
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.05

- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.05

Noise type: *sym* – 40%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.1; $\lambda_u$: 0.1
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.05

Noise type: *sym* – 60%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.1; $\lambda_u$: 0.1
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.05

Noise type: *asym* – 20%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + buffer fit**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **OURs + consolidation**: $lr$: 0.03; $lr_{\text{consolidation}}$: 0.05; $\lambda_u$: 0.005
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05

- **ER + DivideMix**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.01
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.1

Noise type: *asym* – 40%

- **Joint**: $lr$: 0.03
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.03
- **OURs + consolidation**: $lr$: 0.1; $lr_{\text{consolidation}}$: 0.05; $\lambda_u$: 0.1
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.01
- **PuriDivER**: $lr$: 0.001; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.05
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.1

## NTU RGB-D

Noise type: *sym* – 20%

- **Joint**: $lr$: 0.1
- **SGD**: $lr$: 0.1
- **OURs**: $lr$: 0.1
- **OURs + consolidation**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.1; $\lambda_r$: 0.01
- **DividERMix**: $lr$: 0.03
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.3; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.03
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.1
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.3

Noise type: *sym* – 40%

- **Joint**: $lr$: 0.1
- **SGD**: $lr$: 0.03
- **OURs**: $lr$: 0.1
- **OURs + consolidation**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.1; $\lambda_r$: 0.01
- **ER**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05
- **ER + CoTeaching**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **ER + DivideMix**: $lr$: 0.1; $lr_{\text{buffer fit.}}$: 0.05
- **PuriDivER**: $lr$: 0.3; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **PuriDivER.ME**: $lr$: 0.03; $lr_{\text{buffer fit.}}$: 0.05; $\alpha$: 0.1
- **GDumb**: $lr_{\text{buffer fit.}}$: 0.1
- **GDumb + CoTeaching**: $lr_{\text{buffer fit.}}$: 0.1
- **GDumb + DivideMix**: $lr_{\text{buffer fit.}}$: 0.03

## References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, 2019.

[2] Jihwan Bang, Hyunseo Koh, Seulki Park, Hwanjun Song, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on a contaminated data stream with blurry task boundaries. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2022.

[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 2019.

[4] Matteo Boschini, Lorenzo Bonicelli, Pietro Buzzega, Angelo Porrello, and Simone Calderara. Class-incremental continual learning into the extended der-verse. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In *Advances in Neural Information Processing Systems*, 2020.

[6] Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New Insights on Reducing Abrupt Representation Change in Online Continual Learning. In *International Conference on Learning Representations Workshop*, 2022.

[7] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. In *International Conference on Machine Learning Workshop*, 2019.

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2019.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016.

[10] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[11] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A large scale dataset for 3D human activity analysis. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016.

[12] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 2022.