TOWARDS EXPERIENCE REPLAY FOR CLASS-INCREMENTAL LEARNING IN FULLY-BINARY NETWORKS

A PREPRINT

Yanis BASSO-BERT

Univ. Grenoble Alpes, CEA, List F-38000 Grenoble France yanis.bassobert@cea.fr

• Anca MOLNOS

Univ. Grenoble Alpes, CEA, List F-38000 Grenoble France anca.molnos@cea.fr

® Romain LEMAIRE

Univ. Grenoble Alpes, CEA, List F-38000 Grenoble France romain.lemaire@cea.fr

William GUICQUERO

Univ. Grenoble Alpes, CEA, Leti F-38000 Grenoble France william.guicquero@cea.fr

Antoine DUPRET

Univ. Grenoble Alpes, CEA, Leti F-38000 Grenoble France antoine.dupret@cea.fr

March 11, 2025

ABSTRACT

Binary Neural Networks (BNNs) are a promising approach to enable Artificial Neural Network (ANN) implementation on ultra-low power edge devices. Such devices may compute data in highly dynamic environments, in which the classes targeted for inference can evolve or even novel classes may arise, requiring continual learning. Class Incremental Learning (CIL) is a common type of continual learning for classification problems, that has been scarcely addressed in the context of BNNs. Furthermore, most of existing BNNs models are not fully binary, as they require several real-valued network layers, at the input, the output, and for batch normalization. This paper goes a step further, enabling class incremental learning in Fully-Binarized NNs (FBNNs) through four main contributions. We firstly revisit the FBNN design and its training procedure that is suitable to CIL. Secondly, we explore loss balancing, a method to trade-off the performance of past and current classes. Thirdly, we propose a semi-supervised method to pre-train the feature extractor of the FBNN for transferable representations. Fourthly, two conventional CIL methods, i.e., Latent and Native replay, are thoroughly compared. These contributions are exemplified first on the CIFAR100 dataset, before being scaled up to address the CORE50 continual learning benchmark. The final results based on our 3Mb FBNN on CORE50 exhibit at par and better performance than conventional real-valued larger NN models.

Keywords Continual Learning · Class-Incremental Learning · Replay Methods · Binary Neural Network · Quantization

1 Introduction

Binary Neural Networks (BNNs) [1] have emerged as a promising avenue for enabling Artificial Neural Network (ANN) deployment on ultra-low power devices, *e.g.*, always-on processing for computer vision [2, 3, 4, 5, 6]. BNN hardware implementations are highly memory- and energy-efficient with respect to conventional hardware realizations (e.g., floating point or 8-bit integer). Furthermore, BNNs are also compatible with a wide variety of emerging hardware technologies, from custom compute-in-memory to emerging optical processing [7, 8, 9, 10, 11]. Addressing both memory and computational constraints makes BNNs ideal for edge-based inference tasks.

Recently, significant efforts have been made to enhance BNN classification performance through improvements in model structure and training protocols [12, 13]. These advancements have been built upon the assumption that the training

data is identically distributed to the data encountered during inference. This assumption falls short in dynamic scenarios, in which the concepts (*e.g.*, classes) targeted for inference (*e.g.*, classification) can evolve or even novel concepts may arise. Such systems necessitate continual learning [14, 15], *i.e.*, the capability to adapt the model parameters to new concepts without losing the knowledge of old concepts [16].

The Class Incremental Learning (CIL) methods [17][18] address this issue of learning new knowledge while minimizing forgetting in classification problems. A relatively simple and effective method for CIL is Experience Replay (ER) that stores a subset of past experiences in a buffer. Whenever the data of a new class (or set of classes, *i.e.*, task) is available, ER retrains the network on a mix of stored data, interleaved with the new-coming data. ER allows the model to learn from both new and old data, thus mitigating forgetting. Recently, the literature starts to address CIL on BNNs [19, 20]. Nevertheless, to achieve sufficient performance, these approaches still require several real-valued network layers.

To the best of our knowledge, this paper presents the first in-depth study for CIL in Fully Binarized Neural Networks (FBNNs). The main advantage of FBNNs is that their hardware mapping is very efficient. All computations (binarized scalar products with binary inputs) can be performed by highly compact hardware structures, combining point-wise multiplication via logical gates and accumulation using pop-counters [21, 22]. To this end, we made four key contributions, each exemplified in part on the CIFAR100 dataset. Finally, the overall approach is validated on the CORE50 dataset [23].

Firstly, this paper revisits existing work in [24], thoroughly details the salient points to guide FBNN design for CIL, and extend previous results with new metrics and a comparison with state-of-the-art. Most BNNs from the literature are actually not fully binary, with their first and last layers in richer, real-value or integer, representations [25]. Furthermore, to ease convergence, existing BNNs may include Batch Normalization (BN) layers, typically in floating point. To avoid channel-wise BN, we propose to train the network in a single-stage, relying on layer-wise scaling factors only as normalization. The values of the scaling factors are set not from the BN layer parameters, but utilising the dimensions of network topology itself. Furthermore, for the network's bottleneck we propose a trainable global average pooling to reduce dimensionality of the latent space, making the network robust to spatial translations of the input, despite the low expressivity of binary values. Last, to avoid real-valued input data, we transform the input image pixels into a luminance/chrominance domain, which is further converted into a thermometer encoding [26].

Our FBNN is compared with the two closest NN designs, without BN [27, 22] or claiming only binary arithmetic [22]. The performance of FBNNs of three model sizes (parameters' memory footprint) is reported: a 3 Mega bit (Mb), a 9Mb, and a 24Mb. Experiments indicate that our network reaches state-of-the-art performance in an offline setting, *i.e.*, when all the dataset is available. Furthermore, for the CIL experiments, the contributions in this paper mainly focus on our tiny *3Mb-BNN* model. We implement two CIL types of ER, Native replay (storing raw inputs in the buffer) and Latent replay (storing latent features).

Secondly, we address the main CIL trade-off, namely adaptability vs retention, i.e., performance on past tasks vs. on the current task. Loss balancing, a possible strategy to control adaptability and retention, is less addressed in the CIL literature. Along with loss balancing, here we explore three loss functions suitable to BNNs, namely, the conventional Categorical Cross-Entropy (CCE), the squared-hinge, potentially better for BNNs [28], and a focal loss. We find that a CCE loss suits well CIL in FBNN, for both Latent and Native replay. Balancing the loss significantly enhances the final performance of all tasks, while slightly decreasing adaptation.

The proposed FBNN is split in a Feature Extractor (FE) part and a classifier part, as any conventional NN in image classification problems. The common practice for learning feature representation is to perform supervised learning on a pre-training dataset [20]. Extracted features can be discriminative for pre-training classes, however, there is no guarantee that they are also discriminative for later tasks. It becomes an issue for Latent replay, in which the feature extractor is not updated.

Thirdly, the supervised pre-training approach is challenged, proposing a semi-supervised alternative to learn more transferable features in FBNNs. Supervised training is combined with Barlow Twin (BT) [29] and custom activation regularization that forces features to be more distributed. As a results, the final test accuracy in increased by 1.17pts in CIL for the CIFAR100.

Fourthly, the FBNN exploration covers both Native replay and Latent replay. The performance of Native replay and Latent replay are compared for a given memory buffer footprint. For a small memory buffer (<25Mb), Latent replay yields higher final accuracy than Native replay. In this buffer size range, Latent replay offers a better retention but lower adaptation than Native. For larger buffer sizes, Native is better.

Finally, all the proposed methods are put together and experimented on the CORE50 dataset [23]. which serves to benchmark IL strategies in real-life scenarios. To fit this dataset, we design a deeper architecture, *Res-BNN*, following the principles above, but still having only a 3Mb memory footprint. Experiments on CORE50 show that our *3Mb*-

Res-BNN with Latent replay achieves performance comparable to ER methods using a 352Mb ResNet18 [30]. More impressively, Native replay surpasses other replay methods, with a model memory footprint that is $119 \times$ smaller and a buffer size $1.8 \times$ smaller, while relying solely on binary scalar products.

The outline of this paper is as follows. Section 2 discusses the related work for both BNN and CIL. Section 3 states the addressed problem and introduces the necessary background and notations. Section 4 defines the evaluation metrics and methodology. Section 5 introduces the FBNN design guidelines, followed by Section 6, which discusses loss balancing. Section 7 proposes a semi-supervised pre-training of the FE for CIL in FBNN. Next, Section 8 compares Native and Latent replay at iso-memory. Section 9 presents the evaluation of the methods on the CORE50 dataset.

2 BNN and CIL related work

2.1 Binary Neural Networks

Binarization is an extreme case of quantization of NNs, where weights and activations are constraint to 1-bit representation, taking only two values, e.g., $\{+1,1\}$. The standard backpropagation training was made possible for BNNs by the introduction of the Straight-Through-Estimator (STE) [31][32]. Indeed, to train quantized NNs, Quantization Aware Training (QAT) [33] keeps a proxy, full-precision version of the training weights, which are quantized at the forward pass. Whereas BNNs achieved significantly close performance to full precision network on simple datasets, the accuracy gap remains large on challenging ones like ImageNet.

Closing this gap is an ongoing research field. Existing solutions can be categorized as: quantization error mitigation [34], loss function improvement [35], and reducing gradient error [1]. In addition, BNN model enhancements have been proposed more recently with the introduction of skip connections [36], factorized residual block [25] or MUX-based gated residuals [37], this with extensions to RNN [21]. These efforts prioritize classification performance as main figure of merit and while improved accuracy is attained, it often comes at the expense of introducing mixed-precision operations.

Recent approaches further introduce FBNNs, as really hardware-friendly BNNs. Indeed, FBNNs can have a highly efficient implementation with binary-only arithmetic, however they preclude all computation that inherently utilizes real-valued numbers, such as BN. BN-free approaches [27] train BNN by adjusting adaptive gradient clipping and weight scaling. *AB-BNN*[22] improves the offline performance of BN-free networks with a mask layer and adapted ReLU. Furthermore, [38] shows that binarizing the last layer can be done without significant accuracy drop.

This paper extends FBNNs with CIL capabilities. Likewise previous approaches, our work demonstrates that BN can be efficiently replaced by scaling factors, without any significant performance loss. This paper extends [24] with a study of loss balancing and semi-supervised feature extractor pre-training.

2.2 Incremental Learning

Incremental Learning, often also referred to as continual learning, is a paradigm where a network has to learn continually on ever-changing data distribution. An instance of this problem is when a learner is in charge of comprehending a series of tasks only having access to current task data [18].

Several proprieties are desired for such a system: the **adaptation** on current data, the **retention** on past distributions, and, to capitalize on past and current knowledge (forward-transfer and backward-transfer [39]). To avoid an unacceptable model size increase, models should grow significantly sub-linearly with respect to the number of tasks.

For classification problems, [40] defines three scenarios to deal with various data drifts. Class-Incremental Learning (CIL) - addressed in this paper - deals with the occurrence of new concepts (*i.e.*, classes), Domain-Incremental Learning (DIL) with drift in the input data (given a constant number of classes), and Task-Incremental Learning (TIL) is a simplification of CIL where the model knows the task-label.

The solutions to incremental learning problems can be broadly divided into three major categories [17]: regularization-, architecture-, and rehearsal-based method, even thought the core principle remains the same: regularize the current training to prevent forgetting of past knowledge.

Regularization-based methods propose to identify important weights relatively to a prior task and prevent their update by adding penalty terms to the loss [41, 42]. Various "importance" metrics may be utilized. Even thought such methods only need a snap-shot of weights from the previous iteration, in practice they do not scale up to large networks.

Architecture-based methods mitigate catastrophic forgetting by constructing task-specific modules inside the network. It is done via parameter isolation where certain part of the network are frozen [43] and growing architecture where

additional neurons are added to promote adaptation [44]. Such approaches typically require a task-label during prediction.

Rehearsal-based methods mitigate catastrophic forgetting by storing past samples, "replaid" during training. They can be divided into: Examplars-Rehearsal which stores a subset of past examples in a memory buffer [18], and Pseudo-Rehearsal, which uses generative models to synthesize data from past classes in the input [45] or feature space [46]. While Pseudo-Rehearsal reduces memory requirements, it introduces significant challenges for BNNs, especially in generating accurate synthetic binary data. In contrast, Examplars-Rehearsal methods are straightforward to implement, scale well while ensuring stable performance across tasks [17], making it an ideal framework for studying FBNNs in a CIL setting.

The usage of Exemplars-Rehearsal for CIL was first proposed on deep NNs in Incremental Classifier and Representation Learning (iCaRL), which uses herding for selecting exemplars, features knowledge distillation and a nearest-mean classifier. EEiL [47] brings further improvements by utilizing a fully-connected classifier and a balanced fine-tuning stage to correct the bias due to training dataset imbalance. After that, investigations have followed on buffer memory management [48], regularization on gradient [39], knowledge distillation (KD) on logits [49], and KD-induced-bias correction [50].

Moreover, there is a renew interest for the Experience Replay (ER) proposed in [51], which consists in interleaving past examples with the current training batch without additional sophisticated techniques or training protocol. Following [52], we adopt ER as baseline methods for our simulations.

A promising alternative to ER is Latent replay, where latent features are stored instead a raw inputs [53, 54]. In addition to memory overhead reduction, it decreases the training's computational costs, paving the way to on-device model (re)training [55]. Latent replay becomes especially compelling with BNNs, as the latent features are binarized, offering significant storage savings compared to real-values.

2.3 CIL relying on BNN

Lately the literature starts to address CIL on BNNs [19],[20],[56]. Synaptic metaplasticity [56] address catastrophic forgetting in BNN by introducing a metaplasticity-inspired mechanism that modulates weights updates, reducing the likelihood of changes in binary weights that are important for past tasks. This approach lacks scalability, similarly to regularization-based approaches.

More recently, Vorrabi *et al.* propose to adapt Latent replay with a fixed BNN feature extractor and a quantized classifier with CWR* regularization to mitigate forgetting [19]. The work is further extended with more learnable binarized layers, yielding better incremental performance [20]. Nevertheless, this model requires real-valued first, last, and BNs.

Our work proposes to go a step further, addressing CIL in FBNNs, scaled up to the state-of-the-art *CORE50* dataset. In this paper, we discuss the FBNN design and its training procedure, introduce new metrics to analyze the CIL performance, explore loss balancing, detail a method to pre-train the feature extractor for transferable representations, and, finally, thoroughly compare Latent and Native replay.

3 BNN-CIL problem formulation

3.1 CIL problem formulation and notations

In class-incremental learning, a model $f(\theta)$, with parameters θ sequentially learns T classification tasks of training set $D^0,...,D^t,...,D^{T-1}$ respectively. At task t, starting with weights converged on the previous task, f is optimized on current task $D^t = \{(x_i^t, y_i^t)\}_i$ composed of data-labels pairs drawn from a distribution \mathcal{T}^t . An additional dataset D^{PT} is available to pre-train the network. Each task has its own unique set of classes $y_i^t \in \mathcal{Y}_t$ in the considered CIL scenario. For clarity, we denote as offline training the conventional training in which all the data is available at once, when the training is performed. In class-incremental learning, the retraining at each individual task can be considered as offline training on the current task data. For this dataset the samples can be seen several times and there is no epoch nor batch size restrictions.

Furthermore, we consider that task boundaries are given, meaning there is no need to detect new tasks. The objective of the model at task t is to minimize the expected cost on all encountered distributions $\mathcal{T}^i{}_{0 \leq i \leq t}$ as described in (1). However, past distributions are no longer available, we only have access to D^t . Even if the goal is respected at task t-1, after training on D^t the expected cost on past classes is low.

The CIL problem can be formulated as a regularization problem, as follows: how to to regularize training in order to prevent an increase in the loss on past classes without compromising convergence on the current one. The first term

in (1) is the conventional classification loss on the available data of the current task. The second term in (1) is the regularization term that prevents catastrophic forgetting. As the previous data in is not available anymore, this term is approximated, by \mathcal{L}_{ret} , namely the retention loss.

$$\mathcal{L}_{CIL} = \underbrace{\mathbb{E}_{(x,y) \sim \mathcal{T}^t} [\mathcal{L}(f(x,\theta),y)]}_{\text{available data distribution}} + \underbrace{\sum_{i=0}^{t-1} \mathbb{E}_{(x,y) \sim \mathcal{T}^t} [\mathcal{L}(f(x,\theta),y)]}_{\text{approximated by } \mathcal{L}_{ret}}$$
(1)

3.2 Experience Replay (ER)

ER tackles the regularization problem in (1) by keeping a subset of training samples into a bounded memory buffer, referred to as M. At each training session, M is concatenated with D^t to form the training set, and hence M introduces an intrinsic regularization term, as depicted in (2).

$$\mathcal{L}_{CIL} = \underset{(x,y) \sim \mathcal{T}^t}{\mathbb{E}} \left[\mathcal{L}(f(x,\theta),y) \right] + \underbrace{\underset{(x,y) \sim M}{\mathbb{E}} \left[\mathcal{L}(f(x,\theta),y) \right]}_{\mathcal{L}_{ret}}$$
(2)

As such, the CIL system can be described as a *learner* (the BNN) and a *memorizer* (memory buffer). There are four aspects that define an ER strategy: how to update the memorizer, what information to store, for the labels as well as for the data samples, and, finally, how to regularize the loss.

3.2.1 Memorizer update rule

M stores representative samples from each past tasks. In real use cases, the memory budget is bounded. We thus denote as m_s the buffer capacity, in number of samples, and m_b the buffer size, in bits. As in the balanced reservoir sampling strategy [57], we assume that: (1) M should be fully filled to ensure maximum sample diversity within the memory budget, and (2) every class must be equally weighted. Within these assumptions, the memorizer randomly chooses: (1) which samples of the current task to store, and (2) which samples of the buffer to replace, when new samples are stored.

3.2.2 Raw input vs. latent features storing

Two ER approaches are possible: storing native inputs, *i.e.*, **Native replay**, or storing latent features, *i.e.*, **Latent replay**. A FBNN, $f(\theta,.) = G(F(\bullet))$ can be divided into two components: a Feature Extractor (FE), F parameterized by θ_F , and a classifier (Clf), G parameterized by θ_G . The FE corresponds to the early layers and have been shown to be transferable across classification tasks, if the initial training dataset shows enough diversity [58]. Consequently, if F performs dimensionality reduction, storing compressed latent representations in the memory buffer instead of raw inputs can be advantageous. Given a fixed m_b , in this configuration, more samples are stored, making M more diverse but lowering model expressiveness as F cannot be easily updated [54]. The trade-off between diversity and expressiveness is discussed in Section 8.

3.2.3 Labels vs. logits storing

The training set consists of sample-labels pairs. Nevertheless, other input-output mapping can be stored in M. The outputs can be stored as logits or as labels. Typically, the logits is the vector of raw, non-normalized, predictions that the classification model produces (inference outputs). Logits are more informative as they capture classes probability relatively to each other [59]. However, storing logits in CIL requires a bias correction mechanism [50] and it is not clear if they truly enhance performance [60]. Furthermore, storing logits demands more memory than storing labels only, as the logits of each class is an vector of size equal to the number of classes. Everything considered, we choose to store labels as output for all our experiments.

3.2.4 Loss regularization

Several ways to leverage M to regularize the training exist: sample interleaving [57], orthogonal gradient projection [39]. For the sake of simplicity and computational efficiency, we choose the sample interleaving approach where batches are formed by randomly drawing samples in $M \cup D^t$. However, current classes are more represented in D^t than past classes in M, leading to imbalance. Section 6 thoroughly discusses loss balancing.

3.3 The BNN training flow

A binary network, as any quantized network, requires a special procedure for training, such as Quantization Aware Training (QAT). In QAT, the proxy full-precision weights, θ^r , are kept and updated during the training, because binary elements cannot accumulate updates. The forward and backward passes are similar to the ones in conventional training with back-propagation, with few differences, as follows.

Typically the forward pass is performed on the quantized weights. To do so, every full-precision value, x^b , is binarized with the SIGN function (3), resulting in x^b .

$$x^{b} = \operatorname{SIGN}(x^{r}) = \begin{cases} +1, & \text{if } x^{r} \ge 0\\ -1, & \text{otherwise} \end{cases}$$
 (3)

As SIGN is not differentiable, in the backward pass its derivative is approximated by a clipped identity (STE) [31]:

$$\frac{\partial \mathcal{L}}{\partial \theta^r} \simeq \frac{\partial \mathcal{L}}{\partial \theta^b} \mathbb{1}_{[-1;+1]} \tag{4}$$

Special attention should be given to BN [61], an efficient method to ease model convergence by properly scaling the values in forward and backward passes. BN yet relies on full-precision operations that cannot be easily binarized.

A typical QAT multi-stage training [37], [22] is as follows: (1) train with quantized weights, full-precision activations, and BN, (2) train with quantized weights, quantized activations, and BN, and (3) train with quantized weights, quantized activation and scaling factors initialized from BN parameters.

On the last stage, the scaling factors can be computed as an average or median of the estimated moving averages and internal gain parameters. Offsets and biases are discarded as adding values to a binary +1/-1 may flip the sign and introduce large errors. The role of the scaling factors is to help to converge the training when the forward pass is performed only with binary values. The scaling factors are hence removed for inference, since it has no impact on the SIGN function.

4 Metrics and evaluation methodology

In the CIL community, the goal is a high final accuracy, after a given number of learned tasks. Previous work investigates the final accuracy along two directions, adaptation and retention, as defined below. We argue that other metrics allow to better understand the details of a CIL system, as presented in the core of this section. Finally, our experimental setup with respect to CIL and FPNN is extensively detailed.

4.1 Conventional metrics

The three important properties of CIL are the ability to adapt to new classes (adaptation), to remember past classes (retention), and to generalize well on all seen classes (global accuracy). These can be evaluated by the accuracy of the corresponding subset of classes, on the test set. **Adaptation** is measured as the accuracy on the current, *i.e.*, new task D^t , and denoted as a_{new} . **Retention** is measured as the accuracy on the first task D^0 , and denoted as a_{old} . We compute it on D^0 rather than on all past classes $\mathcal{Y}_{i < t}$ to give a indication on the forgetting of the oldest task. **Global accuracy** a_{seen} is the accuracy on all seen classes $\mathcal{Y}_{i \le t}$. For the sake of clarity, we denote as a_{final} the global accuracy after all tasks are learned.

4.2 Beyond conventional metrics

Our learning system comprises a compact FBNN and a bounded memory buffer, both introducing challenges in training. Table 1 summarizes relevant metrics notations. The training accuracy is often omitted in the literature, whereas it offers significant insights on the model expressiveness, providing useful information on overfitting issues. We report the accuracy figures for the train and test cases, when necessary. As the FBNNs studied are compact, a plateau of the training performance may indicate that the test performance is capped by the size of the network, whereas a large training performance may suggest that further regularization could increase test performance.

Furthermore, the buffer size should be low $(e.g., \leq 100 \mathrm{Mb})$, implying a high risk of overfitting. To quantify this, we introduce two metrics: $\mathbf{a}_{\mathrm{buffer}}^{\mathrm{train}}$, the accuracy computed on the buffer samples in the train set, M, and $\mathbf{a}_{\mathrm{buffer}}^{\mathrm{test}}$, the accuracy computed on the test set of past classes $\mathcal{Y}_{i < t}$, at task t.

Table 1: Notations for the accuracy, a, and dispersion, d, metrics. The evaluation is done on test set when the superscript is omitted.

subscript	PT	pre-training task D^{PT}
	old	first task D^0
	new	current task D^t
	seen	all encountered tasks
	final	all tasks
	buffer	samples in buffer M
superscript	train	samples seen during training
	test	samples in the test set

Finally, the network may be initially pre-trained on a dataset D^{PT} and then incrementally retrained. To have an indication of the performance at the start of the incremental retraining we also introduce the accuracy at the pre-training task a_{PT} .

4.3 Dispersion metric

The CIL problem can be framed as a detection problem, instead of a classification problem, where the model's goal is to detect new classes, as well as past classes. The accuracy on a subset of classes only gives the average of classes recall. To provide more insight about each class detection, we propose a dispersion metric, denoted as d, defined as the standard deviation on class recalls. If the model must detect every class equally well, the ideal case is the one where the dispersion is minimal. Combined with the accuracy, dispersion also gives insights on the lower-bound detection performance. Lower accuracy with smaller dispersion can be preferable to higher accuracy with bigger dispersion, if the minimum detection performance is higher. As with accuracy, dispersion can be estimated on a subset of the data, as visible in Table 1.

4.4 Evaluation Methodology

4.4.1 Datasets

In the rest of this paper, the FBNN-CIL is exemplified on CIFAR100 [62]. This is composed of labeled images drawn from 100 classes, each image having 32×32 pixels, with 3 color channels, each encoded on 8-bit integers. Each class is composed of 500 training images and 100 test images. We keep 10% of training images for validation. CIFAR50+5X10 denotes the following scenario: the 50 first classes are used for pre-training $\mathcal{Y}_{PT} = (0, ..., 49)$ and the 50 remaining classes are split into 5 tasks of 10 classes. When possible, the training is regularized using image data augmentation (i.e., random flip, random translation, and random contrast). Section 9 finally reports an in-depth evaluation of the proposed approach on CORE50 [23] with a pre-training on STL10 [63] for a dedicated FBNN (3Mb-Res-BNN).

4.4.2 Hyperparameters

We employ the Adam optimizer, with an adaptive Reduce-On-Plateau learning rate scheduler [64], and an early-stop [65] for each retraining stage. 6 discusses in detail the type of loss, considering Categorical-Cross-Entropy (CCE) as the baseline. Optimization is performed with batch size of 64 and an initial learning rate of 10^{-4} . A small batch size combined to a low initial learning rate were used, as we observed that higher ones may cause training unreliability.

5 Guidelines for FBNN design in CIL

Despite their efficient execution, BNNs are challenging to design and train [66]. The literature proposes various methods for sizing, designing and optimising BNNs [25] [36]. This section revisits and extends related work [24], by stating 5 key design practices to accommodate CIL with ER applied to FBNNs. For the sake of simplicity and repeatability, a basic (but efficient) VGG-like model head 3Mb-BNN (Fig. 1) has been first considered. The model head is composed of 3 convolution blocks, each composed of a stack of two g-grouped convolutions [67] followed by a 2×2 Max Pooling. The number of filters and groups (f,g) are doubled at each block, *i.e.*, (128,1), (256,2) and (512,4). This FE head is followed by a dedicated custom bottleneck and classifier which are described in the following.

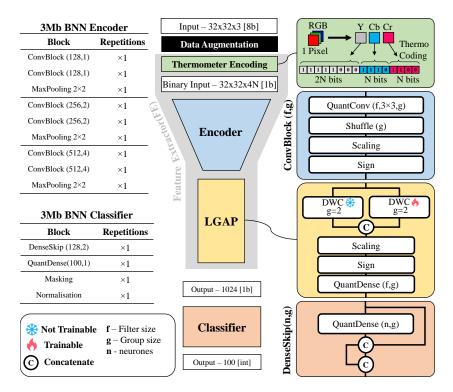


Figure 1: Baseline model structure. 3Mb-BNN version reported here.

5.1 Key BNN enablers for CIL

5.1.1 Scaling factors as normalization

In the offline setting, layer-wise scaling factors can efficiently replace channel-wise BN without performance drop by employing multi-stage QAT. In incremental learning, multi-stage QAT would imply re-initializing BN parameters at the beginning of each retraining. However, BN has been shown to exacerbate catastrophic forgetting by having its parameters, means and variances, biased towards over-represented new classes in training batch. task [68, 69]. Thus it is preferable to not derive scaling factors from the BN parameters in a multi-stage QAT during retrainings.

To circumvent this issue, we propose to train the network in a single-stage, and thus set the scaling factors' values not from the BN layer parameters, but utilising the network topology itself, making them task-agnostic. Inspired by the supposed BN's goal of limiting the covariance shift [61], we aim for scaling factors that give unitary activation variance.

At the initialization, since weights and activations are assumed to be equally distributed between -1 and +1, it means that activation A^b and weights θ^b follow a Rademacher distribution of parameter p=0.5. The resulting dot product between θ^b and A^b thus follows a binomial law of mean 0 and variance $Var=2^2np(1-p)=n$, where n corresponds to number of input of the dot products, also called FanIn. An appropriate scaling factor S could therefore be defined depending on FanIn and a certain scalar K:

$$S = \frac{K}{\sqrt{FanIn}} \tag{5}$$

We conducted experiments to determine the best value for K for the 3Mb-BNN model, so that maximum train accuracy is achieved. Maximum train accuracy is desired to leave room to the network to optimize further by regularization such that good test performance can be also be reached. Table 2 presents the train accuracy for a case with and without BN and for three K values. The unitary variance (K=1) brings comparable results to the use of BN, in a single-stage training.

Special attention should be given to the scaling of the last layer. Experiments indicate that instabilities may occur during training without BN when the number of classes increases. After retraining for several successive tasks, the training loss tends to cap. The reason behind may be that the network is fully binary, and hence the $\{-1,1\}$ representation in the last layer lacks dynamic range, which may result poor output gradient scaling. We find that, what we call an

Table 2: Influence of variance scaling in normalization layer

	BN	K=0.3	K=1	K=3
Train accuracy	78.7%	74.7%	78.5%	75.5%

 α -scaled binarization, *i.e.*, $\alpha \times \text{SIGN}(\dot)$, at the end of the model is beneficial for convergence, as it scales the output gradient. α is dependent on the number of classes, N_c , as follows.

$$\alpha = \frac{1}{\sqrt{c \times FanIn \times N_c}} \tag{6}$$

Similar to the scaling that replaces BN, this scaling is employed during training and removed during inference. The results in Table 2 are on CIFAR100, thus with $N_c=100$, and with the constant c empirically set to 5. The single training procedure is applied at each CIL retraining step. Nevertheless, this single training procedure does not preclude a pre-training stage, when it appears crucial. The weights initialization can be done with any conventional method [70], as well as relying on a proxy pre-training.

5.1.2 Bottleneck design

Including a Global Average Pooling layer (GAP) at the end of the convolutions blocks is a common practice to both reduce the dimensionality of latent representation before the classifier and make the network robust to spatial translation of the input [71]. However, with binary features as input, the importance of a given spatial region cannot be encoded thought the activation magnitude. To keep spatial information at the output of the FE we propose to add a path where the pooling weights are learned. The whole projection block is noted LGAP (learnable GAP) and depicted in Fig. 1. In practice, binary weighting is learned as a Depth Wise Convolution of the spatial size of input filters. Both projections are concatenated, scaled, and binarized. Finally, features are recombined by a dense layer in a 1024 dimension binary latent space. It results in a compression factor of 24 compared to the input 8-bit image.

5.1.3 Input data encoding

A common practice in BNN design is to keep the first layer in full-precision at the cost of its computational overhead. Pixel normalization, centering and standardization are thus used to pre-process the input. We aim for binary-only arithmetic for each layer, comprising the first and last ones. [26] proposes to code each RGB channel with a thermometer coding, denoted as TRGB, on 85 binary channels after a uniform quantization. This leads to better performance than RGB with pixel normalization. Each pixel in an image thus requires $3 \times log_2(\lceil 85 \rceil) = 21$ bits of storage in the case of ER with a buffer.

In a fully binary implementation, the hardware mapping can be even more eased when the number of bits per pixel are an integer power of 2. To further optimize the input data encoding, RGB values are first transformed into a YCbCr (YCC) color space and then converted into a thermometer coding, TYCC. Since the luminance channel, Y, contains more discriminative information that the chrominance channels, CbCr, we allocate 2 times more channels to the Y (2N) than to the C channels (N). Experimentally we determined that TYCC with N=16 C channels gives the same performance as TRGB for CIFAR-100 in offline training, and, in addition, it requires 25% less number of bits per pixel. In the rest of this paper we hence utilize TYCC with N=16.

5.1.4 Classifier depth

We consider that the classifier begins after the LGAP block. In Latent replay, this defines the frontier between trainable and frozen layers. The question that arises is: how deep the classifier should be? On the one hand, many approaches choose a single dense layer [72] for a classifier. Hence the FE output must be linearly separable and thus it is easy to analyze. Finally, a single dense layer eases the optimization as a stack of dense layers is harder to optimize. On the other hand, in case of Latent replay, a single layer does not leave a lot of room for adaptation to future tasks. We propose to use a multi-layer classifier with skip connections as depicted Fig. 1 with the DenseSkip block with n neurons and g groups. In a full precision dense layer, skip connections provide shortcuts ensuring the classifier will perform at least as well as a linear one. Here, binary weights values do not allow null projections and thus do not allow path gating. To that end, we duplicate the input activations, so that the 0 value can be obtained by opposite sign weighting on the same activation node. Finally, grouped dense layers (with 2 groups) are introduced to cap the classifier model size.

5.2 FBNN evaluation

This section first evaluates the proposed FBNN 3Mb-BNN in an offline setting to put in perspective the performance in comparison with state-of-the-art BNNs, as well as with their full-precision counterparts. Second, we define and evaluate two CIL baselines, that represent the lower- and upper-bound performance for any incremental learning strategy.

5.2.1 Baselines

In the offline setting, in order to assess the effect of binarization, we evaluate our 3Mb-BNN against two floating-point variants: (1) a memory-footprint equivalent network, $FPNN_b$, and (2) a topology equivalent network $FPNN_p$. Therefore, $FPNN_b$ and $FPNN_p$ have real-valued weights and activations in a floating-point (e.g., considered coded on 32b). On the one hand, $FPNN_b$ keeps the same topology as our 3Mb-BNN but has $32 \times$ less parameters. The number of parameters reduction is done by scaling down the number of filters. For a fair comparison, $FPNN_b$ uses BN and conventional input normalization between [-1, +1] (no thermometer coding). On the other hand, $FPNN_p$ has the same architecture and the same number of parameters as 3Mb-BNN, and hence a $32 \times$ larger memory footprint while using BN. This evaluation (Fig. 2) comprises three network sizings: 3 Mb (TINP), 9 Mb (TINP), 9 Mb (TINP), and 24 Mb (TINP). Fig. 1 presents TINP0 Mb and 24 Mb FBNNs are variants obtained by multiplying respectively by 2 and 4 the number of filters. TINP1 Size also compared to TIPP2 and TIPP3 and TIPP4 How training approaches without BN but with input and output layers using full-precision. TIPP3 And TIPP4 Restauctured results both rely on using two different backbones: BiResNet18 [36] (11.7 Mb) and ReActNetA [73] (29.3 Mb).

In the CIL setting, we evaluate our model on the CIFAR50+5X10 scenario on four experience replay baselines: *naive*, *naive-reset*, *cumulative* and *cumulative-reset*. The *naive* baselines correspond to the case of retraining only with the new task data, without any incremental learning strategy, hence highlighting catastrophic forgetting; they are thus equivalent to experience-replay with an empty buffer. The *naive* baselines offer a lower bound on the performance of any incremental learning strategy. The *cumulative* baselines correspond to the case of an infinite memory buffer, where all past labelled data are stored, hence indicating the upper-bound performance. The *-reset* suffix indicates that weights are randomly initialized at the beginning of each retraining. The *reset* baselines assess the influence of parameter initialization.

5.2.2 Offline comparison: binary vs. 32-bits floating-point

Fig. 2 reports offline accuracy as a function of the model size (Mb), indicating that 3Mb-BNN provides better performance than $FPNN_b$ on train set and test set for the 3 Mb and 9 Mb configurations. We also note that the 24 Mb variant does not improve test accuracy compared to the 9 Mb, but it still improves the training accuracy. This plateau where increasing the model size above 10-Mb does not lead to better test accuracy, is common [25]. Large BNNs seem more difficult to regularize. $FPNN_p$ models achieve higher test accuracy by a bit more than 10% compared to our FBNNs, while training accuracy remains capped to 100% (confirming results of [66]).

When compared to existing BNNs without BN, our 3Mb-BNN model shows better performance than [27] with more than $3 \times$ less parameters, and similar performance to [22] in the Large-BNN configuration. In conclusion, our FBNN reaches state-of-the-art performance in this particular offline setting.

In all the experiments of this paper we only consider 3Mb-BNN, unless explicitly stated otherwise. Moreover, since the training accuracy does not cap to 100%, the 3Mb-BNN is the most challenging candidate to study compact FBNN for CIL. It particularly allows to evaluate the forgetting issues in the context of a very limited expressiveness of the model.

5.2.3 CIL baselines analysis

Fig. 3 reports the accuracy, for the four baselines, on each task, per epoch, during the whole CIFAR50+5X10 incremental dataset, for *3Mb-BNN*. Table 3 complements the results with metrics at the end of all 5 retrainings. When looking at the validation curve on the current task dataset, we can notice that our training protocol enables a robust adaptation at each task and for each baseline.

As expected, the *naive* baselines retain only the task of the last retraining, whereas in the *cumulative* baselines all tasks are learned. For each task, t, the validation accuracy on the current task D^t is higher for *naive* than *cumulative*. For *cumulative*, the training includes every past task's training set, $D_0, ..., D_t$, whereas *naive* includes only the current task, D_t . Therefore, simultaneous learning of multiple tasks does not contribute to improve learning performance for individual tasks. Resetting the weights in the *naive* baseline has little influence on the performance; the network forgets all the tasks except the last one regardless on how the weights are initialized.

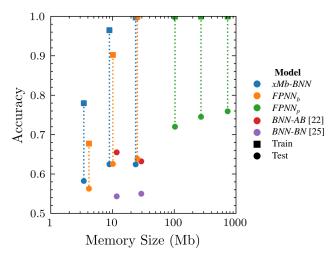


Figure 2: Offline accuracy for different configurations of our architecture. Test accuracy is reported with circle and train accuracy with squares.

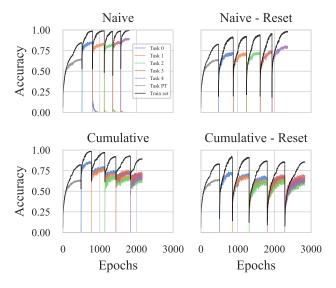


Figure 3: Training curves of *3Mb-BNN* on the CIL CIFAR50+5X10, for the 4 baselines strategies. Accuracy on train set is reported on black, pre-training test accuracy in grey, and the test accuracy of each task in a different colors.

On the cumulative baselines, weight initialization has a crucial impact. When the weights are initialized with their value from the previous training, the network achieves both higher accuracy on current and old tasks. The same result is observed in Table 3. a_{new} and a_{old} are significantly higher without *reset*. This indicates that *3Mb-BNN* benefits from weight initialization from the previous task, to increase performance on following tasks. At the end of all tasks, the *cumulative* setting has a 3pts increase in final accuracy compared to the *cumulative-reset* setting. Moreover, the number of epochs is reduced by 41% for the entire retraining for the 5 tasks when the weights are not reset. It indicates our FBNNs capitalize on past knowledge to better learn new tasks.

6 Loss balancing

In general, we can consider that each incremental training session with a replay buffer is a training on an imbalanced dataset containing all seen classes $D^t \cup M$. Typically, there are more samples per class in the dataset of the new task t, D^t , than in the buffer with past task's samples, M. The common practice to compensate for an imbalanced dataset is to weight the loss. However, CIL introduces two additional biases that can challenge this common practice. The first

baseline	a_{old}	d_{old}	a _{new}	d_{new}	$a_{\rm final}$	d_{final}	epochs
naive	0.0%	0	89.8%	0.06	17.9%	0.36	236
naive-reset	0.0%	_	79.7%				428
	71.0%	1	l				318
cumulative-reset	66.0%	0.14	62.9%	0.13	63.9%	0.14	545

Table 3: Task-level metrics during last retraining (Task 4).

bias concerns weight initialization at each retraining, which is biased toward past classes. The second bias concerns pre-training of model weights in Latent replay. Indeed, some real-life use-cases may require to retain the classes in the pre-training task. As the feature extractor is trained on D^{PT} then fixed, there is a strong bias toward the retention of the pre-training task.

As methods to deal with these two biases, in what follows we investigate: (1) the influence of inverse-frequency loss class-balancing, (2) the utilization of a focal term, and (3) the choice of the classification loss. These three factors are studied for two use-cases, Retained Pre-training Task (RPT) and Forgotten Pre-training Task (FPT), which, as the name suggests, deal with remembering of not the pre-training task.

Losses and loss-balancing strategies

Inverse class-frequency adjustment is a common strategy to solve the class-balancing problem. The loss terms associated to training samples of class $i \in \{1, ..., C\}$ are weighted by a factor w_i computed as defined in (7). We note the classcardinality and the total number of samples in the re-training dataset, $D^t \cup M$, as n_i and N, respectively. f_i additionally denotes the frequency of a class i as in (8).

We evaluate the inverse class-frequency weighting for three losses: Categorical Cross-Entropy (CCE), Focal Categorical Cross-Entropy (FCCE), and Squared-Hinge (SH).

$$w_{i} = C \times \frac{(f_{i})^{-1}}{\sum_{j=1}^{C} (f_{j})^{-1}}$$

$$f_{i} = \frac{n_{i}}{N}$$
(7)

$$f_i = \frac{n_i}{N} \tag{8}$$

The CCE is conventionally defined as in (9) where B is the batch size, $y_{i,c}$ the one-hot encoded label for sample i and class c and $p_{i,c}$ the predicted probability for class c for sample i obtained after applying *softmax* function to logits.

$$\mathcal{L}_{CCE} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{C} w_{c}.y_{i,c}.log(p_{i,c})$$
(9)

The FCCE loss [74] is a variant of the standard categorical cross-entropy loss that is designed to address class imbalance by down-weighting the loss assigned to well-classified examples, focusing more on harder-to-classify examples. FCCE introduces a tunable focusing parameter ν that adjusts how much attention is paid to difficult examples. (10) defines FCCE using the same notation as above:

$$\mathcal{L}_{FCCE} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{C} w_c (1 - p_{i,c})^{\nu} y_{i,c} . log(p_{i,c})$$
(10)

The SH loss, designed for Support Vector Machines (SVM), has shown promising results with BNN [28]. Moreover, as many CIL methods use margin loss for classification [17], we consider it in the loss comparison for BNN-CIL. SH penalizes miss-classifications with a quadratic cost, leading to a smoother gradient and encouraging larger margins between classes for better classification performance. (11) defines SH where the label $y_{i,c}$ is in $\{-1,+1\}$ (1 if the sample belongs to class c, -1 otherwise) and $\hat{y}_{i,c}$ is the associated logits.

$$\mathcal{L}_{SH} = \frac{1}{B} \sum_{i=1}^{B} \sum_{c=1}^{C} w_c \cdot \left(\max\left(0, 1 - y_{i,c}^{\hat{}} \cdot y_{i,c}\right) \right)^2$$
 (11)

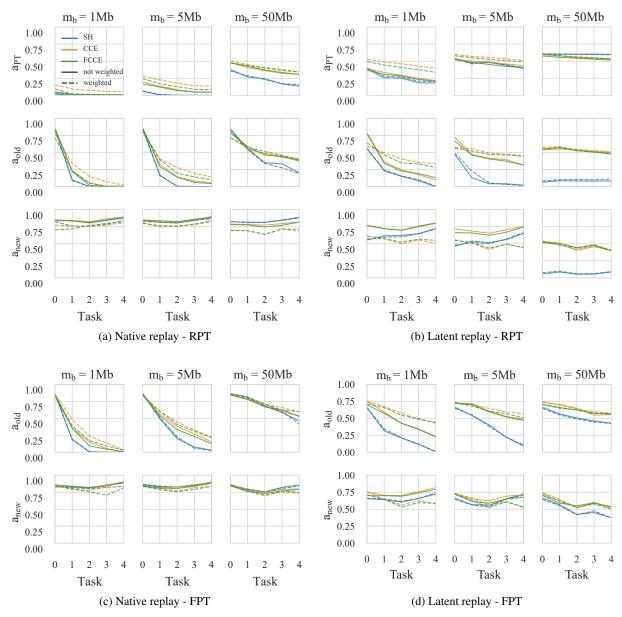


Figure 4: Influence of loss and buffer size in RPT and FPT scenarios. Latent replay and Native replay strategies are reported with various losses (colors).

6.2 Experimental results

3Mb-BNN is evaluated on CIFAR50+5X10 for both Native and Latent replay. Unlike FPT, for RPT, the classifier is not reset after pre-training and the memory buffer includes samples from D^{PT} . As in [74], we set $\nu=2$ for FCCE loss.

6.2.1 The SH loss

With respect to Native replay, Fig. 4a and 4c indicate that SH leads to better performance on the current task, $a_{\rm new}$, than other losses at the expense of deteriorating retention, $a_{\rm old}$, regardless of the buffer size. SH seems to exacerbate plasticity to the point of degrading the final accuracy. Figures also indicate that SH is relatively insensitive to class-weighting, which is probably because, by definition, it attempts to find margins under a one-vs-all classification problem statement.

strategy	loss	weighted		not weighted	
		$a_{\rm final}$	d_{final}	$a_{\rm final}$	d_{final}
Latent (5Mb)	SH	28.0%	0.30	27.4%	0.29
	CCE	45.5%	0.18	40.9%	0.22
	FCCE	44.7%	0.16	38.7%	0.21
Native (50Mb)	SH	23.5%	0.28	22.3%	0.29
	CCE	40.2%	0.19	38.1%	0.23
	FCCE	38.9%	0.19	37.5%	0.22

Table 4: Comparison of balancing strategies in the RPT scenario.

Table 5: Comparison of balancing strategies in the FPT scenario.

strategy	loss	weighted		not wei	ighted	
		a _{final}	d	$a_{\rm final}$	d	
Latent (5Mb)	SH	25.6%	0.28	25.6%	0.29	
	CCE	51.6%	0.19	48.6%	0.22	
	FCCE	48.7%	0.17	47.1%	0.19	
Native (50Mb)	SH	48.8%	0.28	48.6%	0.28	
	CCE	57.5%	0.19	56.5%	0.21	
	FCCE	57.5%	0.19	54.5%	0.20	

In Latent replay, even with a 50Mb buffer, the SH loss exhibits convergence issues on current tasks while the accuracy on the pre-training stays high.

6.2.2 The cross-entropy losses

Let's first analyze the adaptation and retention capacity in Fig. 4. As expected, inverse frequency balancing improves retention, i.e., $a_{\rm old}$, for both Native and Latent replay, especially with smaller buffers. By giving higher weight w_i to the under-represented past classes, the model preserves knowledge from earlier tasks, enhancing retention. However, this comes at the cost of reduced adaptation to new classes, i.e., $a_{\rm new}$, as the increased focus on past data limits the model's ability to adjust to new information. Latent replay is more sensitive to balancing. For instance, we can observe a 41.0% reduction in adaptation on the last task for Latent replay compared to a 7.7% reduction for Native replay with CCE loss in RPT scenario m_b =5 Mb.

Secondly, the final performance is shown in Table 4 and Table 5 for RPT and FPT. For clarity, we report final metrics for Latent 5 Mb and Native 50 Mb, *i.e.*, a similar buffer capacity in terms of number of samples. In both configurations, approximately 50 samples per class are stored. In both configurations, loss balancing leads to improvement of a_{final} . Particularly in the RPT scenario, loss balancing increases the final performance by at least +5pts, in Latent replay (5Mb) for both losses. Weighting the loss decreases class dispersion by a factor 9.7% in average for FPT scenario and by 18.2% in RPT scenario. Loss balancing is thus effective at both enhancing final accuracy and reducing class dispersion in both scenarios.

6.2.3 Loss focalization

The focal loss does not seem to bring any substantial advantage in the final accuracy (Table 5 and 4). However, it reduces classification dispersion at the cost of small final accuracy decrease. If the use-case where the CIL is employed requires that each task is classified with (more or less) the same accuracy, one can consider utilizing FCCE.

6.2.4 Main takeaways

Despite being a good practice in offline learning in BNN, the SH loss is not highly relevant for FBNN-CIL as it decreases the final accuracy. A CCE loss suits well FBNN-CIL. In this case, balancing the loss significantly enhances performance in every configuration while slightly compromising the adaptation. *i.e.*, decreasing the accuracy on the new task. The dispersion in per-class performance can be reduced by adding a focal term. Latent replay appears more sensitive to the type of loss especially on a small buffer (*e.g.*, 1Mb) than Native replay. In the rest of this paper all experiments use the CCE loss with inverse-frequency class-weighting. In the following sections we focus on FPT, where the pre-training only learns the FE.

7 Semi-supervised pre-training of the FE

A common practice for learning feature representation is to perform supervised learning on a pre-training task, with data D^{PT} . In this manner, extracted features are dimiscrinative for the pre-training classes, however, there is no guarantee that they are also discriminative for the tasks that are incrementally learned after the pre-training, named downstream tasks. This is particularly problematic for conventional Latent replay, in which the feature extractor is not updated. In this section we challenge the common fully supervised pre-training approach, elaborating an approach to learn richer, transferable features. We first discuss Self-Supervised Learning (SSL) as a promising task-agnostic alternative. We then propose an activation regularization term that promotes features diversity. Finally we evaluate the influence of both on the CIFAR50+5x10 dataset.

7.0.1 Self-Supervised Regularization

Learning representation with self-supervision can boost performance in CIL settings [75][30]. Introducing a proxy task, such as clustering [76], rotation prediction [77], contrastive loss [78], or maximizing the information content of the embedding [29], makes the feature representation more task-agnostic. SSL thus prevents FE to over-specialize on a given task.

To validate the interest of SSL for BNN pre-training, we adapt the state-of-the-art Barlow Twin (BT) loss [29]. SSL on BNN is a recent research field and early work [79] has highlighted the difficulties to adapt contrastive approaches, *i.e.*, MOCO-V2 [78], without complex distillation loss and multi-phase training. Furthermore, existing results are obtained on larger BNN architecture with BN layers. To be compatible with our compact FBNN with single-phase training we look for self-supervised approach that adds minimal mechanisms. The BT approach achieves state-of-the-art performance with minimal computational overhead. Unlike other methods, the BT has the advantage to work with small batch size, which decreases the chance of training issues in FBNN.

The overall approach is described Fig. 5. Two augmentations X_1 and X_2 of the same image batch X are projected with a projector block after FE. We denote as $\mathcal C$ the correlation matrix between the two projections. The self-supervision proxy task consists in minimizing the distance between $\mathcal C$ and the identity matrix. We can decompose the loss into two terms, as presented in (12). The first one encourages the diagonal components to be close to 1 enforcing feature invariance to data augmentation and the second one minimizes off-diagonal terms preventing features redundancy. Both terms are weighted with $\lambda=10^{-5}$, as in the original paper.

$$\mathcal{L}_{SSL} = \sum_{i} (1 - \mathcal{C}_{ii})^2 + \lambda \sum_{i} \sum_{j \neq i} \mathcal{C}_{ij}^2$$
(12)

7.0.2 Features regularization

Since binary latent features seem highly correlated by classes, we believe that forcing latent features to be more distributed (less correlated) in the binary latent space could help capturing more diverse characteristics. We thus propose a regularization term \mathcal{L}_{FR} that promotes components to have opposite binary value within a batch of samples. As described in (13), this activation regularization loss acting on z minimizes the component's sum along the batch axis B, for each of the D latent features.

$$\mathcal{L}_{FR} = \frac{1}{D \times B} \sum_{d=1}^{D} \sum_{b=1}^{B} z_{b,d}$$
 (13)

7.0.3 Multi-objective weighted loss

SSL and latent features regularization adds two proxy tasks that require no label, which we believe is key to learn more transferable features. However, during pre-training labels are available. Moreover, Section 6 indicates that that supervised learning with CCE is a good baseline for representation learning. We therefore propose a combination of both supervised and self-supervised learning, and hence the loss minimization is formulated as a multi-objective optimization problem. \mathcal{L}_{CCE} , \mathcal{L}_{SSL} and \mathcal{L}_{r} are then combined in a single loss with α , β and γ weights:

$$\mathcal{L}_{PT} = \alpha \mathcal{L}_{CCE} + \beta \mathcal{L}_{SSL} + \gamma \mathcal{L}_{FR} \tag{14}$$

The proposed pre-training loss is computed as described in Fig. 5. \mathcal{L}_{CCE} and \mathcal{L}_{FR} are computed using the previously mentioned data augmentation T_p (Section 4.4) on each input batch X. Since \mathcal{L}_{SSL} requires a stronger data augmentation,

paradigm	α	β	γ	${ m a_{PT}^{train}}$	$a_{\mathrm{PT}}^{\mathrm{test}}$	d_{PT}	${ m a_{final}^{train}}$	$a_{\rm final}^{\rm test}$	d_{final}
supervised	1	0	0	90.74%	62.90%	0.15	56.33%	51.28%	0.17
regularization	1	0	10^{-3}	90.82%	64.06%	0.14	55.78%	51.86%	0.18
	1	0	10^{-2}	90.52%	63.26%	0.14	55.58%	51.96%	0.19
	1	0	10^{-1}	86.85%	61.04%	0.14	51.97%	48.62%	0.19
SSL	1	10^{-6}	0	90.09%	64.16%	0.13	56.47%	51.08%	0.18
	1	10^{-5}	0	89.57%	63.58%	0.14	57.63%	51.92%	0.18
	1	10^{-4}	0	86.12%	60.34%	0.16	57.88%	51.26%	0.18
	1	10^{-3}	0	63.84%	48.40%	0.17	50.04%	41.76%	0.19
regularization +SSL	1	5.10^{-6}	5.10^{-3}	90.24%	63.54%	0.14	56.53%	52.34%	0.20
	1	10^{-5}	10^{-2}	90.02%	63.51%	0.14	57.43%	52.45%	0.19

Table 6: Final retraining performance of CIFAR50+5X10 using Latent replay (10Mb buffer) for various regularization setups.

we thus added random color transformations and random crop, denoted as T_c . As the pre-training can be done without compute resource constraints, the projection block is composed of the stack of two full-precision dense layers with 2048 neurons followed by a BN and a ReLU activation.

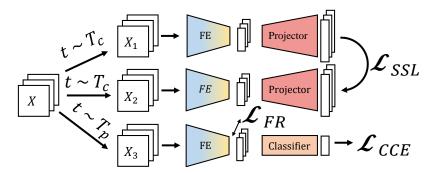


Figure 5: Illustration of the multi-objective pre-training framework.

7.1 Experimental Results

Loss terms compromises are evaluated on *CIFAR50+5X10* for Latent replay, a 10Mb memory buffer, the *3Mb-BNN* model and a class-balanced CCE. Table 6 reports metrics for the end of the pre-training stage and at the end of the 5 retrainings.

7.1.1 The SSL term

The BT loss effectively improves the final performance for $\beta=10^{-5}$, for both train and test accuracy (+1.3pts and +0.7pts). Hence, the representation learned with SSL is better suited to downstream tasks. Indeed, SSL eases the convergence on new tasks, boosting generalization on the test set. The SSL term also enhances the test accuracy on the pre-training task, $a_{\rm PT}^{\rm test}$, while reducing the train accuracy, $a_{\rm PT}^{\rm train}$ for $\beta \leq 10^{-5}$. This observation indicates that the SSL term forces the representation to be less discriminative for the training set and it helps regularizing the network.

7.1.2 The latent features regularization

Feature regularization improves both pre-training test accuracy and final test accuracy for $\gamma \le 10^{-2}$ by 1.16pts and 0.58pts. A too strong regularization impairs both initial and final performances.

However, contrary to \mathcal{L}_{SSL} , \mathcal{L}_{FR} improves the pretraining accuracy and reduces the final accuracy on train set. It indicates that the loss helps the optimization on the pre-training task but not on downstream tasks. This can be alleviated by combining both SSL and feature regularization loss with best performance reached with $\beta=10^{-5}$ and $\gamma=10^{-2}$ increasing final test accuracy by 1.17pts. Note that the best configuration (α,β,γ) tested appears to be close to the optimum reached for β and γ when used independently.

7.1.3 Main takeaways

This experiment underlines the importance of the pre-training quality and its influence on the CIL performance. Alternative learning paradigms, as self-supervised learning are promising solutions to learn more transferable features without considerable additional costs.

8 Native vs. Latent replay

This section compares the performance of Native replay and Latent replay for a given memory footprint, comparison also referred to as 'at iso-memory'. The qualitative trade-offs between these two approaches are summarized in Table 7.

In our specific configuration, Latent replay allows to store $13 \times$ more samples per Mb than Native. Therefore, Latent replay potentially approximates better past distributions, at iso-memory. This advantage becomes particularly prominent as the number of classes increases. Latent replay also offers computational cost reduction, with 90% less parameters to update and less iterations to convergence (see Section 5).

However, this Latent replay gain in number of samples in the buffer is obtained at the cost of a decrease in model reconfigurability. The latent representation can not be updated, otherwise the samples in the buffer would not be representative anymore of the image classes. In particular, this problem is referred to in the literature as the aging problem.

Another non-negligible advantage of Native replay is the possibility to augment the data, which is vital for compact networks. Data augmentation is particularly needed on the memory buffer as only a few samples are kept. However, contrary to Native replay, in Latent replay there is no conventional way to do data augmentation from a binary latent space.

Native		Latent
-	#Sample/Mb	++
+	Information/sample	-
-	Computational cost	+++
+	Data Augmentation	_
+	Pre-training dependency	_

Table 7: Pros (+) and cons (-) of Native versus Latent replay.

Fig. 6 presents the impact of the replay buffer size on the final performance for Native and Latent replay with CIFAR50+5X10: (a) $a_{\rm seen}$ represents the final performance on all seen classes, (b) $a_{\rm new}$ represents the adaptation capacity, *i.e.*, on new training data (without buffer samples), and (c) $a_{\rm buffer}$ represents the retention capacity, *i.e.*, on past classes. The previously presented loss balancing and pre-training are applied. To exhibit the influence of the buffer size on the FE training policy, the Latent and Native strategies are evaluated for buffer sizes ranging from 0 to the entire dataset.

8.0.1 Final performance (a)

We observe that, under 25Mb, Latent replay offers the best performance. Above 25Mb, Native replay gives better performance. One can note that the 25Mb point corresponds to the upper bound for latent memory buffer, *i.e.*, all the training dataset can be stored, it brings no improvement to Latent replay to have a larger memory size.

8.0.2 Adaptation (b)

When we consider the performance on new classes, i.e., $a_{\rm new}$ accuracy, two distinct interpretations of replay regularization can be made. For latent replay, the augmentation of m_b decrease both train and test performance by the same magnitude (approximately 30% from 1 sample/image to full dataset). Optimization on new task gets harder. On the contrary, for Native replay only the test accuracy decreases. The FBNN still has the ability to model the new training distribution but it has lost its ability to generalize on it.

8.0.3 Retention (c)

When looking at the test accuracy on old classes, *i.e.*, dotted lines, we first note, as expected, a sharp increase in retention for larger buffer size for both strategy. However, when the FE is non trainable, the performance decreases on a larger buffer, highlighting the limited modeling capacity. On the contrary, a trainable FE maintains high accuracy across

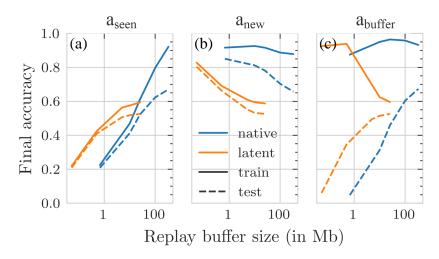


Figure 6: Iso-memory comparison between Native and Latent replay.

the buffer size, on buffer samples. We nevertheless observe a slight decrease for small buffers, likely explained by the strong imbalance in the training set $D^t \cup M$.

8.0.4 Main takeaways

The choice to let the FE trainable depends on the memory budget. For a small memory buffer (<25Mb), Latent replay yields higher final accuracy than Native. In this case, Latent offers a better retention but lower adaptation than Native. If obtaining the best performance on the current task at the cost of less retention is preferred, the FE can be trainable, regardless of the buffer size.

9 Evaluation on the CORE50 dataset

In this section, we evaluate our approach developed in the previous sections with a more realistic benchmark, the CORE50 dataset [23]. Recently introduced in the CIL community, CORE50 serves to benchmark strategies in real-life scenarios. CORE50 comprises 128×128 pixels images belonging to 50 classes of objects acquired across 11 recording sessions, with 300 consecutive frames (images) per session.

We adopt the scenario of [30] to compare our approach to prior work. It consists in 10 retraining tasks of 5 classes obtained by randomly splitting the dataset. Sessions #3,#7, and #10 represent the test dataset and the others the train dataset. Compared to *CIFAR100*, *CORE50* enables us to extend our conclusions to larger, more complex images, with temporal correlations during acquisition and under various backgrounds.

A deeper architecture, 3Mb-Res-BNN, is designed to comply with CORE50 requirements, adhering to the previously stated principles and remaining within the 3Mb model memory size budget. We subsequently evaluate 3Mb-Res-BNN Latent and Native replay against existing approaches. As there is no pre-training task in the CORE50 scenario, we propose to pre-train our 3Mb-Res-BNN model on the STL10 dataset [63].

This is a well-know benchmark for evaluating unsupervised and semi-supervised learning algorithms. It consists of 10 classes, including various object categories. The dataset features 5000 labeled training 96×96 color images (500 images/class) and 100000 unlabeled images.

9.1 Deeper FE with skip-connections

3Mb-Res-BNN accommodates larger images thanks to a deeper architecture composed of residual blocks, as depicted in Fig. 7. After a first down-scaling with a convolution block of stride 2, we stacked residual blocks that preserve the spacial feature size, and down-sampling blocks that halve the spatial feature size, inspired by ShuffleNetv2 [67]. For the residual block, the main and residual inputs paths are obtained by splitting the feature map in half in the channel dimension. The main path consists in a stack of 1×1 , 3×3 , and 1×1 convolution blocks with a higher number of groups on the 3×3 convolutions. The down-sampling blocks are similar to the res-blocks but with a stride of on 3×3 convolutions. The residual path contains a 3×3 and 1×1 convolution block. In both blocks, the path aggregation is

done by concatenation, as opposed to addition which could lead to bit overflow. After, feature maps are shuffled with a shuffle layer of group 2.

When putting it all together, 3Mb-Res-BNN results in a 3M parameters architecture with 17 blocks and a 1024 dimensions latent space. With this FE, Latent replay therefore allows to store $80 \times$ more examples than Native, at iso-memory.

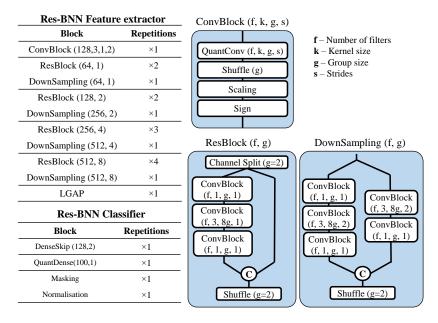


Figure 7: 3Mb-Res-BNN architecture - Deeper FE to comply with CORE50.

9.2 Experimental Results

Table 8: Evaluation metrics on CORE50 Benchmark. A buffer of 50 samples is used for TA and 100 samples for TF.

Method	CORE50-TA	CORE50-TF
ER [51] (reported from [30])	41.72 ± 1.30	21.80 ± 0.70
ER-Aug	44.16 ± 2.05	25.34 ± 0.74
DER++ [49] (reported from [30])	46.62 ± 0.46	22.84 ± 0.84
DER++-Aug	45.12 ± 0.68	28.10 ± 0.80
CTN [80] (reported from [30])	54.17 ± 0.85	N/A
CTN-Aug	53.40 ± 1.37	N/A
DualNet [30]	57.64 ± 1.36	38.76 ± 1.52
3Mb-Res-BNN Latent replay (ours)	53.96 ± 1.17	17.22 ± 0.16
3Mb-Res-BNN Native replay (ours)	59.07 ± 1.24	44.52 ± 0.54

The STL-10 pre-training consists of a multi-phase training protocol with BN, to start from a better weight initialization. The first phase consists in a standard supervised training only on labeled images only. BN layers are then replaced by scaling-factors, as introduced in Section 5.1. Then, a second training phase is performed on the entire dataset, benefiting from unlabelled data for the unsupervised training loss terms. \mathcal{L}_{CCE} is computed on labeled images and \mathcal{L}_{FR} and \mathcal{L}_{SSL} on unlabelled images, for regularization. The training hyperparameters are kept the same as in the previous sections.

The comparison to prior work is rather difficult, as few approaches investigate exactly the same CIL scenario. Nevertheless, we compare our approach to continual learning approaches with a replay buffer [30]. The baselines are ER [51], DER++ [49], an ER variant with knowledge distillation on logits, CTN [80] a TIL approach with fixed FE and DualNet [30] a CIL method that uses SSL to slowly update the FE. The first three methods are evaluated (Table 8) with or without data augmentation (-Aug suffix). The network used in these articles relies on ResNet18 [81] with a 352Mb memory size, *i.e.*, more than $100 \times$ larger than 3Mb-Res-BNN. Unlike our approach, in [30] the CIL is optimized on 3 epochs with a very high learning rate of 0.03. However, 3Mb-Res-BNN requires smaller learning rate, *i.e.*, 10^{-4} ,

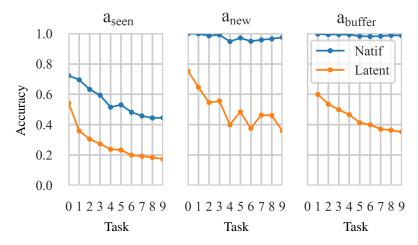


Figure 8: Task-Free (TF) CIL performance for 3Mb-Res-BNN in Native and Latent replay on seen,new and buffer subset. 100 samples per class are stored for Native replay and we use an equivalent buffer size for Latent replay.

(as empirically observed, higher learning rates, $> 10^{-3}$, may cause training instability) and hence a larger number of epochs (approximately a hundred) is necessary for convergence.

Table 8 reports the final accuracy as proposed in the literature, without task-label (Task-Free, TF) and with task-label (Task-Aware, TA), respectively corresponding to CIL and TIL. We report the final training accuracy for Native replay with 100 images per classes for TF setting and 50 per classes for TA setting, similarly to the other methods in [30]. In the same manner, we report accuracy with Latent replay with an equivalent memory buffer size.

In the TF case, 3Mb-Res-BNN with Latent replay is slightly below the range of performance of the state-of-the-art ER methods without relying on any floating point operation and with $100 \times$ less parameters. Nevertheless, in TA setting, 3Mb-Res-BNN Latent replay outperforms ER, DER and CTN with its 53.96% final accuracy. 3Mb-Res-BNN with Native replay presents state-of-the-art performance in TF with 44.52%, which is far better than DualNet, despite of using a $100\times$ smaller model memory footprint, a $7.4\times$ smaller buffer footprint thanks to our TYCC-16 image encoding and binary-only arithmetic that tremendously reduces the associated computational complexity.

We further investigate incremental training dynamics in Fig. 8 (TF). With Native replay, 3Mb-Res-BNN succeeds to maintain a high level of adaptation with more than 90% on new task training sets. It also fully captures the knowledge in the buffer with perfect classification score on buffer samples. On the contrary, with Latent replay, 3Mb-Res-BNN's ability to fit the training data (new task and buffer) diminishes through retraining from 70% to 37%. The classifier alone has not the capacity to fit the whole classification problem. However, we can conclude that the FE can still extract meaningful features while having been trained on a completely different dataset. Indeed on the first CIL task, Latent replay is already 12%pts below Native replay while having a fixed FE.

10 Conclusion

This study presents an exploration of Fully Binarized Neural Networks (FBNNs) within the context of Class Incremental Learning (CIL), particularly through the lens of Experience Replay (ER). By addressing critical challenges such as network design, loss balancing, and pre-training, we have demonstrated the viability of FBNNs in dynamic environments where computing resources are constrained. Our findings emphasize the importance of tailored network architectures for BNN-CIL, which are crucial for maintaining expressiveness, mitigating convergence issues and keeping a small memory buffer. The integration of carefully designed loss functions and pre-training strategies enhances adaptability and reduces catastrophic forgetting. Furthermore, our experiments highlight the trade-offs between Native and Latent replay methods, with Latent replay proving to be more effective in memory-constrained settings. Finally, our CORE50 results shows that for a challenging benchmark, a careful design of a high-end FBNN model (3Mb-Res-BNN) as well as its incremental training protocol, can even outperform other CIL frameworks that rely on full-precision network models.

Acknowledgments

This work is part of the IPCEI Microelectronics and Connectivity and was supported by the French Public Authorities within the frame of France 2030.

References

- [1] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, September 2020.
- [2] Arnaud Verdant, William Guicquero, Nicolas Royer, Guillaume Moritz, Sébastien Martin, Florent Lepin, Sylvain Choisnet, Fabrice Guellec, Benoît Deschamps, Sylvain Clerc, and Jérôme Chossat. A 3.0μW@5fps QQVGA Self-Controlled Wake-Up Imager with On-Chip Motion Detection, Auto-Exposure and Object Recognition. In 2020 IEEE Symposium on VLSI Circuits, pages 1–2, June 2020.
- [3] Sumon Kumar Bose and Arindam Basu. A 389 TOPS/W, Always ON Region Proposal Integrated Circuit Using In-Memory Computing in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 58(2):554–568, February 2023.
- [4] David Garrett, Youn Sung Park, Seongjong Kim, Jay Sharma, Wenbin Huang, Majid Shaghaghi, Vinay Parthasarathy, Stephen Gibellini, Stephen Bailey, Mallik Moturi, Pieter Vorenkamp, Kurt Busch, Jeremy Holleman, Behrooz Javid, Alireza Yousefi, Mohsen Judy, and Atul Gupta. A 1mW Always-on Computer Vision Deep Learning Neural Decision Processor. In 2023 IEEE International Solid-State Circuits Conference (ISSCC), pages 8–10, February 2023.
- [5] Arnaud Verdant, William Guicquero, David Coriat, Guillaume Moritz, Nicolas Royer, Sébastien Thuries, Anaïs Mollard, Vincent Teil, Yann Deprez, Gilles Monnot, Pierre Malinge, Bruno Paille, Guillaume Caubit, Arnaud Bourge, Laurent Tardif, Stéphanie Bigault, and Jérôme Chossat. A 450μW@50fps Wake-Up Module Featuring Auto-Bracketed 3-Scale Log-Corrected Pattern Recognition and Motion Detection in a 1.5Mpix 8T Global Shutter Imager. In 2024 IEEE Symposium on VLSI Circuits, pages 1–2, June 2024.
- [6] Xueyong Zhang and Arindam Basu. A 915–1220 TOPS/W, 976–1301 GOPS Hybrid In-Memory Computing Based Always-On Image Processing for Neuromorphic Vision Sensors. *IEEE Journal of Solid-State Circuits*, 58(3):589–599, March 2023.
- [7] Karam Cho, Akul Malhotra, and Sumeet Kumar Gupta. XNOR-VSH: A Valley-Spin Hall Effect-Based Compact and Energy-Efficient Synaptic Crossbar Array for Binary Neural Networks. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 9(2):99–107, December 2023.
- [8] Hao Cai, Zhongjian Bian, Yaoru Hou, Yongliang Zhou, Jia-le Cui, Yanan Guo, Xiaoyun Tian, Bo Liu, Xin Si, Zhen Wang, Jun Yang, and Weiwei Shan. A 28nm 2Mb STT-MRAM Computing-in-Memory Macro with a Refined Bit-Cell and 22.4 41.5TOPS/W for AI Inference. In 2023 IEEE International Solid-State Circuits Conference (ISSCC), pages 500–502, February 2023.
- [9] Khaled Alhaj Ali, Amer Baghdadi, Elsa Dupraz, Mathieu Léonardon, Mostafa Rizk, and Jean-Philippe Diguet. MOL-Based In-Memory Computing of Binary Neural Networks. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 30(7):869–880, July 2022.
- [10] Taha Shahroodi, Raphael Cardoso, Mahdi Zahedi, Stephan Wong, Alberto Bosio, Ian O'Connor, and Said Hamdioui. Lightspeed Binary Neural Networks using Optical Phase-Change Materials. In 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 1–2, April 2023.
- [11] Febin P. Sunny, Asif Mirza, Mahdi Nikdast, and Sudeep Pasricha. ROBIN: A Robust Optical Binary Neural Network Accelerator. *ACM Trans. Embed. Comput. Syst.*, 20(5s):57:1–57:24, September 2021.
- [12] Chunyu Yuan and Sos S. Agaian. A comprehensive review of Binary Neural Network. *Artificial Intelligence Review*, 56(11):12949–13013, November 2023.
- [13] Ratshih Sayed, Haytham Azmi, Heba Shawkey, A. H. Khalil, and Mohamed Refky. A Systematic Literature Review on Binary Neural Networks. *IEEE Access*, 11:27546–27578, 2023.
- [14] Ramesh Kumar Sah, Seyed Iman Mirzadeh, and Hassan Ghasemzadeh. Continual Learning for Activity Recognition. In 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), pages 2416–2420, July 2022.
- [15] Amrutha Machireddy, Ranganath Krishnan, Nilesh Ahuja, and Omesh Tickoo. Continual Active Adaptation to Evolving Distributional Shifts. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 3443–3449, June 2022.

- [16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017.
- [17] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-Incremental Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2024.
- [18] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5533–5542, 2017.
- [19] Lorenzo Vorabbi, Davide Maltoni, and Stefano Santi. On-device learning with binary neural networks. In *Image Analysis and Processing ICIAP 2023 Workshops: Udine, Italy, September 11–15, 2023, Proceedings, Part I*, page 39–50, Berlin, Heidelberg, 2024. Springer-Verlag.
- [20] Lorenzo Vorabbi, Davide Maltoni, Guido Borghi, and Stefano Santi. Enabling On-Device Continual Learning with Binary Neural Networks and Latent Replay:. In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 25–36, Rome, Italy, 2024. SCITEPRESS.
- [21] Van Thien Nguyen, William Guicquero, and Gilles Sicard. BILLNET: A Binarized Conv3D-LSTM Network with Logic-gated residual architecture for hardware-efficient video inference. In 2022 IEEE Workshop on Signal Processing Systems (SiPS), pages 1–6, November 2022.
- [22] Ruichen Ma, Guanchao Qiao, Yian Liu, Liwei Meng, Ning Ning, Yang Liu, and Shaogang Hu. A&B BNN: Add&Bit-operation-only hardware-friendly binary neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2024.
- [23] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR, 13–15 Nov 2017.
- [24] Yanis Basso-Bert, William Guicquero, Anca Molnos, Romain Lemaire, and Antoine Dupret. On class-incremental learning for fully binarized convolutional neural networks. In 2024 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–5. IEEE, 2024.
- [25] Tom Bannink, Arash Bakhtiari, Adam Hillier, Lukas Geiger, Tim de Bruin, Leon Overweel, Jelmer Neeven, and Koen Helwegen. Larq compute engine: Design, benchmark, and deploy state-of-the-art binarized neural networks. *ArXiv*, abs/2011.09398, 2020.
- [26] Daniel Bankman, Lita Yang, Bert Moons, Marian Verhelst, and Boris Murmann. An Always-On 3.8 μJ86% CIFAR-10 Mixed-Signal Binary CNN Processor With All Memory on Chip in 28-nm CMOS. *IEEE Journal of Solid-State Circuits*, 54(1):158–172, 2019.
- [27] Tianlong Chen, Zhenyu Zhang, Xu Ouyang, Zechun Liu, Zhiqiang Shen, and Zhangyang Wang. "bnn bn = ?": Training binary neural networks without batch normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4619–4629, June 2021.
- [28] Van Thien Nguyen, William Guicquero, and Gilles Sicard. A 1Mb Mixed-Precision Quantized Encoder for Image Classification and Patch-Based Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(8):5581–5594, 2022.
- [29] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.
- [30] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34, 2021.
- [31] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, abs/1308.3432, 2013.
- [32] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 29, 2016.
- [33] George K. Thiruvathukal, Yung-Hsiang Lu, Jaeyoun Kim, Yiran Chen, Bo Chen, Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference. pages 291–326, January 2022. Book Title: Low-Power Computer Vision Edition: 1 Publisher: Chapman and Hall/CRC.

- [34] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.
- [35] Lu Hou, Quanming Yao, and James Tin-Yau Kwok. Loss-aware binarization of deep networks. *ArXiv*, abs/1611.01600, 2016.
- [36] Zechun Liu, Wenhan Luo, Baoyuan Wu, Xin Yang, W. Liu, and K. Cheng. Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision*, 128:202 219, 2018.
- [37] Van Thien Nguyen, William Guicquero, and Gilles Sicard. Histogram-Equalized Quantization for logic-gated Residual Neural Networks. In 2022 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1289–1293, May 2022.
- [38] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- [39] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc.
- [40] Gido M. van de Ven and Andreas Savas Tolias. Three scenarios for continual learning. ArXiv, abs/1904.07734, 2019.
- [41] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- [42] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.
- [43] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.
- [44] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.
- [45] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.
- [46] Miguel Solinas, Marina Reyboz, Stephane Rousset, Julie Galliere, Marion Mainsant, Yannick Bourrier, Anca Molnos, and Martial Mermillod. On the beneficial effects of reinjections for continual learning. *SN Comput. Sci.*, 4(1):37, 2023.
- [47] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [48] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019.
- [49] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.
- [50] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Raymond Fu. Large scale incremental learning. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 374–382, 2019.
- [51] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97 2:285–308, 1990.
- [52] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. 2020 25th International Conference on Pattern Recognition (ICPR), pages 2180–2187, 2020.
- [53] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. page 699–715, Berlin, Heidelberg, 2020. Springer-Verlag.

- [54] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 10203–10209, 2020.
- [55] Leonardo Ravaglia, Manuele Rusci, Davide Nadalini, Alessandro Capotondi, Francesco Conti, and Luca Benini. A tinyml platform for on-device continual learning with quantized latent replays. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):789–802, 2021.
- [56] Axel Laborieux, Maxence Ernoult, Tifenn Hirtzlin, and Damien Querlioz. Synaptic metaplasticity in binarized neural networks. *Nature Communications*, 12(1):2549, May 2021.
- [57] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019.
- [58] Eden Belouadah and Adrian Daniel Popescu. Deesil: Deep-shallow incremental learning. In ECCV Workshops, 2018.
- [59] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3513–3522, June 2021.
- [60] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021.
- [61] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pages 448–456, 07–09 Jul 2015.
- [62] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [63] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.
- [64] François Chollet et al. Keras, 2015.
- [65] Lutz Prechelt. Early stopping-but when? In Neural Networks: Tricks of the trade, pages 55–69. Springer, 2002.
- [66] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.
- [67] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 6848–6856, jun 2018.
- [68] Sungmin Cha, Sungjun Cho, Dasol Hwang, Sunwon Hong, Moontae Lee, and Taesup Moon. Rebalancing batch normalization for exemplar-based class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20127–20136, 2023.
- [69] Vincenzo Lomonaco, Davide Maltoni, Lorenzo Pellegrini, et al. Rehearsal-free continual learning over small non-iid batches. In *CVPR Workshops*, volume 1, page 3, 2020.
- [70] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
- [71] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. CoRR, abs/1312.4400, 2013.
- [72] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.
- [73] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 143–159. Springer, 2020.
- [74] T-YLPG Ross and GKHP Dollár. Focal loss for dense object detection. In proceedings of the IEEE conference on computer vision and pattern recognition, pages 2980–2988, 2017.
- [75] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5867–5876, 2021.
- [76] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. 2020.

- [77] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. ArXiv. abs/1803.07728, 2018.
- [78] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.
- [79] Zhiqiang Shen, Zechun Liu, Jie Qin, Lei Huang, Kwang-Ting Cheng, and Marios Savvides. S2-bnn: Bridging the gap between self-supervised real and 1-bit neural networks via guided distribution calibration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2165–2174, 2021.
- [80] Quang Pham, Chenghao Liu, Doyen Sahoo, and Hoi Steven. Contextual transformation networks for online continual learning. In *International Conference on Learning Representations*, 2021.
- [81] 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*, pages 770–778, 2016.