Citation for published version:
Banerjee, S, Verma, VK & Namboodiri, VP 2023, Streaming LifeLong Learning With Any-Time Inference. in *Proceedings - ICRA 2023: IEEE International Conference on Robotics and Automation.* Proceedings - IEEE International Conference on Robotics and Automation, vol. 2023-May, IEEE, U. S. A., pp. 9486-9492, 2023 IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK United Kingdom, 29/05/23. https://doi.org/10.1109/ICRA48891.2023.10160358

DOI:
10.1109/ICRA48891.2023.10160358

Publication date:
2023

Document Version
Peer reviewed version

Link to publication

Publisher Rights
CC BY-ND

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Mar. 2026

Streaming LifeLong Learning With Any-Time Inference

Soumya Banerjee[1], Vinay Kumar Verma[2], and Vinay P. Namboodiri[3]

***Abstract*— Despite rapid advancements in the lifelong learning (LL) research, a large body of research mainly focuses on improving the performance in the existing *static* continual learning (CL) setups. These methods lack the ability to succeed in a rapidly changing *dynamic* environment, where an AI agent needs to quickly learn new instances in a 'single pass' from the non-i.i.d (also possibly temporally contiguous/coherent) data streams without suffering from catastrophic forgetting. For practical applicability, we propose a novel lifelong learning approach, which is streaming, i.e., a single input sample arrives in each time step. Moreover, the proposed approach is single pass, class-incremental, and is subject to be evaluated at any moment. To address this challenging setup and various evaluation protocols, we propose a Bayesian framework, that enables fast parameter update, given a single training example, and enables any-time inference. We additionally propose an implicit regularizer in the form of snap-shot self-distillation, which effectively minimizes the forgetting further. We further propose an effective method that efficiently selects a subset of samples for online memory rehearsal and employs a new replay buffer management scheme that significantly boosts the overall performance. Our empirical evaluations and ablations demonstrate that the proposed method outperforms the prior works by large margins.**

I. INTRODUCTION

In this paper, we aim to achieve 'any-time-inference' ability in *streaming lifelong learning* (S-LL). This is an appealing ability as it allows practical applicability of AI agents/robots, which can be continually updated and deployed without catastrophic forgetting [1]. While deep learning models can achieve state-of-the-art performance throughout various tasks ranging across vision, language, speech, etc., these models suffer from catastrophic forgetting [2], if trained incrementally with sequentially arriving data. LifeLong Learning (LL) or continual learning (CL) [3] studies the problem of training a deep neural network (DNN) continuously with sequentially coming data without forgetting. This paper introduces a novel approach, which enables DNN for class-incremental streaming classification. *Class-incremental streaming classification/learning* (CISL) is a challenging variant of LL [4], [5], with the following essential and desirable properties:

i) Learner sees each labeled instance only once (*desirable*),
ii) Learner needs to adapt the newly available data immediately (*desirable*),
iii) The input data stream can be non-i.i.d and temporally contiguous/coherent (*essential*),
iv) The AI agent can be evaluated at any moment, i.e., 'any-time-inference' (*essential*),

[1] IIT Kanpur, India, soumyab@cse.iitk.ac.in
[2] Duke University, USA, vinaykumar.verma@duke.edu
[3] University Of Bath, UK, vpn22@bath.ac.uk

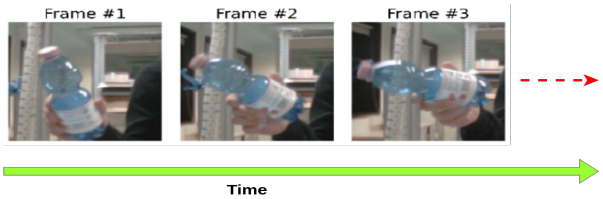

Fig. 1: S-LL addresses the problem of learning continuously from a stream of temporally correlated (non-i.i.d) labeled data, for e.g., learning to recognize an object from multiple viewpoints with no forgetting. In this fig., 3 temporally correlated frames of bottle are taken from iCubWorld 1.0 [13].

v) The AI agent should be able to predict a class label among all the observed classes so far during inference without having the task-id [6] (*essential*),
vi) For practical applicability, learner must limit their memory usage (*desirable*).

A wide variety of CL approaches [3], [7], [8], [9], [10], [11], [12] have been proposed to alleviate catastrophic forgetting in DNN. However, most of the existing approaches mainly focus on 'incremental-batch-learning' (IBL) [3], where the input data in each incremental step arrives in batches, and the agent visits them multiple times in order to enable CL. While these methods are applicable in a static environment, these methods are ill-suited for a rapidly changing dynamic environment, where an AI agent requires to learn from the newly available knowledge immediately with no forgetting. Different from this, 'streaming learning' (S-LL) imposes a unique set of restrictions/challenges (as mentioned earlier) and has not received widespread attention like the other existing CL approaches [4], [5]. However, its utility is apparent, as it enables the practical deployment of autonomous AI agents. Hayes et al. [5] argued that S-LL is closer to biological learning than other existing CL scenarios. Fig. 1 demonstrates an example of streaming classification scenario. In Table I, we classify the existing CL approaches in accordance to the underlying assumption that they impose. It can be observed that ExStream [4] and REMIND [5] are the only two CL approaches, that focus on streaming classification. However, it is worth mentioning that ExStream [4] uses full buffer replay, which violates the subset buffer replay constraint of CISL. Furthermore, while REMIND [5] satisfies all the CISL constraints, it stores a comparably large number of past samples in memory, which limits its applicability.

In this work, we overcome the aforementioned limitations in a principled manner and introduce a novel CISL method, dubbed as **Ba**yesian Class Incremental **S**tream**i**ng **L**earning (BaSiL), that seeks to incorporate the ability of rehearsal with a functional regularization into a simple yet

TABLE I: Categorization of the baseline approaches depending on the underlying simplifying assumptions they impose. In $\zeta(n)$, n represents the number of gradient steps required to train the corresponding model. $\zeta(n) \gg \zeta(1)$. '-' indicates, that we are unable to find the exact value.

				'Class-Incremental Streaming Learning' (CISL) Crucial Properties									
Methods	Type	Bayesian Framework	Batch-Size (N_t)	Fine-tunes	Single Pass Learning	CIL	Subset Buffer Replay	Training Time	Inference Time	Violates Any *CISL* Constraint	Memory Capacity	Regularization Based	Memory Based
EWC [3]	Batch	✗	$N_t \gg 1$	✗	✗	✗	n/a	$\zeta(n)$	$\zeta(1)$	✓	n/a	✓	✗
MAS [7]	Batch	✗	$N_t \gg 1$	✗	✗	✗	n/a	$\zeta(n)$	$\zeta(1)$	✓	n/a	✓	✗
VCL [8]	Batch	✓	$N_t \gg 1$	✗	✗	✗	n/a	$\zeta(n)$	$\zeta(1)$	✓	n/a	✓	✗
Coreset VCL [8]	Batch	✓	$N_t \gg 1$	✓	✗	✗	✗	$\zeta(n)$	$\zeta(n)$	✓	-	✓	✓
GDumb [14]	Online	✗	$N_t \gg 1$	✓	✗	✓	✗	$\zeta(1)$	$\zeta(n)$	✓	-	✗	✓
TinyER [15]	Online	✗	$N_t \gg 1$	✗	✓	✓	✓	$\zeta(1)$	$\zeta(1)$	✗	$\leq 5\%$	✗	✓
DER [16]	Online	✗	$N_t \gg 1$	✗	✓	✓	✓	$\zeta(1)$	$\zeta(1)$	✗	$\leq 5\%$	✓	✗
DER++ [16]	Online	✗	$N_t \gg 1$	✗	✓	✓	✓	$\zeta(1)$	$\zeta(1)$	✗	$\leq 5\%$	✓	✓
ExStream [4]	Streaming	✗	$N_t = 1$	✗	✓	✓	✗	$\zeta(1)$	$\zeta(1)$	✓	$\leq 5\%$	✗	✓
REMIND [5]	Streaming	✗	$N_t = 1$	✗	✓	✓	✓	$\zeta(1)$	$\zeta(1)$	✗	$\gg 10\%$	✗	✓
Ours	Streaming	✓	$N_t = 1$	✗	✓	✓	✓	$\zeta(1)$	$\zeta(1)$	✗	$\leq 5\%$	✓	✓
CISL Constraints			$N_t = 1$	✗	✓	✓	✓	$\zeta(1)$	$\zeta(1)$				

effective framework of streaming variational Bayes. We leverage both the replay samples and the sequential data to compute the joint likelihood in order to approximate the posterior in each incremental step. As a consequence, the proposed approach does not require explicit finetuning unlike approaches like [14], and can be evaluated 'on-the-fly' at any moment. We also propose a novel online 'loss-aware' buffer management policy and various sample selection strategies that include the most informative sample(s) in memory or select for rehearsal, which significantly boosts the model's performance. Our experimental results on three benchmark datasets demonstrate the effectiveness of our proposed method to circumvent forgetting in highly challenging streaming scenarios.

Our contributions can be summarized as follows:

- We introduce BaSiL, a novel rehearsal-based dual regularization framework, which includes a streaming Bayesian framework and a functional regularizer, to alleviate forgetting in challenging CISL setup,
- We propose a novel online 'loss-aware' replay buffer management policy and various sampling strategies which significantly boosts the model's performance,
- We empirically show that the proposed method can be evaluated at any time without requiring to finetune unlike [14], [8] with no significant drop in final accuracy,
- Empirical evaluations and ablations on three benchmark datasets validate the superiority of BaSiL over the existing baselines.

II. RELATED WORK

Parameter-isolation-based approaches train different subsets of model parameters on sequential tasks. PNN [17], DEN [18] expand the network to accommodate the new task. PathNet [19], PackNet [12], Piggyback [20], and HAT [21] train different subsets of network parameters on each task.

Regularization-based approaches use an extra regularization term in the loss function to enable continual learning. LWF [22] uses knowledge distillation [23] loss to prevent forgetting. EWC [3], IMM [24], SI [11] and MAS [7] regularize by penalizing changes to the important weights of the network.

Rehearsal-based approaches [25], [26] replay a subset of past training data during sequential learning. iCaRL [10], SER [27], and TinyER [15] use memory replay when training on a new task. DER/DER++ [16] uses knowledge distillation [23] and memory replay while learning a new task. DGR [28], VGR [29], MeRGAN [30], and CloGAN [9] uses generative models to replay past samples during CL.

VCL [8] uses Bayesian inference to mitigate forgetting [2]. However, the approach, when naïvely adapted, performs poorly in S-LL. Additionally, it requires task-id during inference. Furthermore, explicit finetuning with the coreset samples restricts it from performing any time inference.

REMIND [5] is a recently proposed rehearsal-based S-LL approach, which follows a setting close to the one proposed. However, it stores comparatively a large number of buffer samples, for e.g., iCaRL [10] stores only 10K samples for ImageNet [31] experiment, whereas REMIND stores 1M samples. Furthermore, it uses a lossy compression to store past samples. While it is an interesting approach, it is complementary to our proposed technique. We believe that we make a more substantial algorithmic contributions.

III. PROPOSED APPROACH

A. Streaming Learning With A Single Example

In the following, we introduce BaSiL, which trains a convolutional neural network (CNN) in CISL setup. Formally, we separate the CNN into two networks (Fig. 2): (i) non-plastic feature extractor $G(\cdot)$, containing the initial U layers of CNN, and (ii) plastic neural network $F(\cdot)$, containing the final V layers. For a given input image $\boldsymbol{x}$, the predicted class label is computed as: $y = F(G(\boldsymbol{x}))$. We initialize the parameters of feature extractor $G(\cdot)$ with the pre-trained weights and keep it frozen throughout S-LL. We model the plastic network $F(\cdot)$ as a *Bayesian-neural-network* (BNN) [32], [33] and optimize its parameters with the sequentially coming data in CISL setup without forgetting.

In the below, we describe how the plastic network $F(\cdot)$ (BNN) is optimized with a single posterior approximation in each incremental step, t, with a single data point: $\boldsymbol{D}_t = \{d_t\} = \{(\boldsymbol{x}_t, y_t)\}$, arriving sequentially in a streaming manner with no forgetting [2].

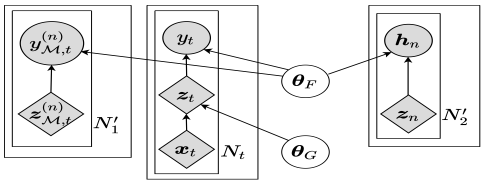

Fig. 2: Schematic representation of the proposed model. $\boldsymbol{\theta}_G$ denotes the parameters of the frozen feature extractor $G(\cdot)$, whereas $\boldsymbol{\theta}_F$ denotes the parameters of BNN $F(\cdot)$.

Variational Posterior Estimation. Streaming learning naturally emerges from the Bayes' rule [34]; given the posterior $p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t-1})$, whenever a new data: $\boldsymbol{D}_t = \{d_t\} = \{(\boldsymbol{x}_t, y_t)\}$ arrives, the new posterior $p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t})$ can be computed by combining the previous posterior and the new data likelihood, i.e., $p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t}) \propto p(\boldsymbol{D}_t|\boldsymbol{\theta})\; p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t-1})$, where the old posterior is treated as the prior. However, for any complex model, the exact Bayesian inference is not tractable, and an approximation is needed. A Bayesian neural network (BNN) [32], [35] commonly approximates the posterior with a variational posterior $q(\boldsymbol{\theta})$ by minimizing the KL divergence (in Eq. 1), or equivalently by maximizing the evidence lower bound (ELBO) (in Eq. 2):

$$\boldsymbol{L}_t(\boldsymbol{\theta}) = \underset{q\in\mathcal{Q}}{\arg\min}\, \mathrm{KL}\left(q_t(\boldsymbol{\theta})\,||\,\frac{1}{Z_t}q_{t-1}(\boldsymbol{\theta})p(\boldsymbol{D}_t|\boldsymbol{\theta})\right) \tag{1}$$

$$\backsimeq \underset{q\in\mathcal{Q}}{\arg\max}\, \mathbb{E}_{\boldsymbol{\theta}\sim q_t(\boldsymbol{\theta})}\left[\log p(\boldsymbol{D}_t|\boldsymbol{\theta})\right] - \mathrm{KL}\left(q_t(\boldsymbol{\theta})\,||\,q_{t-1}(\boldsymbol{\theta})\right) \tag{2}$$

KL-divergence in Eq. 2 works as an implicit parameter regularizer, by keeping the prior and the posterior distribution close to each other, which can minimize forgetting in sequential learning. However, there are a few concerns. Firstly, optimizing the ELBO (in Eq. 2) to approximate the posterior $p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t})$ with only a single training example, i.e., $\boldsymbol{D}_t = \{d_t\} = \{(\boldsymbol{x}_t, y_t)\}$, in each incremental step, t, can fail in S-LL [36]. Furthermore, likelihood estimation from a single example: $\boldsymbol{D}_t = \{d_t\} = \{(\boldsymbol{x}_t, y_t)\}$ can bias the model towards the new data disproportionately and can maximize the confusion during inference in class incremental learning (CIL) setup [6].

We, therefore, propose to estimate the likelihood from both the new example: $\boldsymbol{D}_t$ and the previously observed examples in order to approximate the new posterior: $p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t})$. For this purpose, a small fraction ($\leq 5\%$) of past observed examples are stored as representatives in a *fixed-sized* tiny episodic memory buffer $\boldsymbol{M}$. However, instead of storing raw pixels (images) $\boldsymbol{x}$, we store the embedding $\boldsymbol{z} = G(\boldsymbol{x})$, where $\boldsymbol{z} \in \mathbb{R}^d$. This allows us to avoid the cost of recomputing the image-embeddings and expedites the learning even further. It also saves a significant amount of space and allows us to keep more examples in a small budget.

During training on the new example: $\boldsymbol{D}_t = \{d_t\} = \{(\boldsymbol{x}_t, y_t)\}$, we select a subset of samples: $\boldsymbol{D}_{\boldsymbol{M},t}$ from the memory $\boldsymbol{M}$, where: $(i)\boldsymbol{D}_{\boldsymbol{M},t} \subset \boldsymbol{M}$, $(ii)|\boldsymbol{D}_{\boldsymbol{M},t}| = N'_1 \ll |\boldsymbol{M}|$, *instead of replaying the whole buffer*, and compute the likelihood jointly with the new example $\boldsymbol{D}_t$ to estimate the new posterior. We, therefore, compute the new posterior as follows: $p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t}) \propto p(\boldsymbol{D}_t|\boldsymbol{\theta})\; p(\boldsymbol{D}_{\boldsymbol{M},t}|\boldsymbol{\theta})\; p(\boldsymbol{\theta}|\boldsymbol{D}_{1:t-1})$. Since the exact Bayesian inference is intractable, we approximate it with a variational posterior $q_t(\boldsymbol{\theta})$ as follows:

$$\boldsymbol{L}^1_t(\boldsymbol{\theta}) = \underset{q\in\mathcal{Q}}{\arg\min}\, \mathrm{KL}\left(q_t(\boldsymbol{\theta})\,||\,\frac{1}{Z_t}q_{t-1}(\boldsymbol{\theta})p(\boldsymbol{D}_t|\boldsymbol{\theta})p(\boldsymbol{D}_{\boldsymbol{M},t}|\boldsymbol{\theta})\right) \tag{3}$$

Note that Eq. 3 consists of two likelihood terms, that is, one for the new data and other for the subset buffer replay. It minimizes the confusion in the network parameters while predicting a class label over all the observed classes in CIL setup. Further, it allows the model to be evaluated 'on-the-fly' at any moment.

The above minimization (in Eq. 3) can be equivalently written as the maximization of the evidence lower bound (ELBO) as follow:

$$\begin{aligned}\boldsymbol{L}^1_t(\boldsymbol{\theta}) = {} & \mathbb{E}_{\boldsymbol{\theta}\sim q_t(\boldsymbol{\theta})}\left[\log p(y_t|\boldsymbol{\theta}, G(\boldsymbol{x}_t))\right] \\ & + \sum_{n=1}^{N'_1} \mathbb{E}_{\boldsymbol{\theta}\sim q_t(\boldsymbol{\theta})}\left[\log p(y^{(n)}_{\boldsymbol{M},t}|\boldsymbol{\theta}, \boldsymbol{z}^{(n)}_{\boldsymbol{M},t})\right] \\ & - \lambda_1 \cdot \mathrm{KL}\left(q_t(\boldsymbol{\theta})\,||\,q_{t-1}(\boldsymbol{\theta})\right)\end{aligned} \tag{4}$$

where: (i) $\boldsymbol{D}_t = \{d_t\} = \{(\boldsymbol{x}_t, y_t)\}$, (ii) $\boldsymbol{D}_{\boldsymbol{M},t} = \{d^{(n)}_{\boldsymbol{M},t}\}^{N'_1}_{n=1} = \{(\boldsymbol{z}^{(n)}_{\boldsymbol{M},t}, y^{(n)}_{\boldsymbol{M},t})\}^{N'_1}_{n=1}$, (iii) $\boldsymbol{D}_{\boldsymbol{M},t} \subset \boldsymbol{M}$, (iv) $|\boldsymbol{D}_{\boldsymbol{M},t}| = N'_1 \ll |\boldsymbol{M}|$, and (v) λ_1 is a hyper-parameter.

Snap-Shot Self Distillation. It is worth noting that KL divergence minimization (in Eq. 4) between prior and posterior distribution works as an inherent parameter regularizer, and minimizes the changes in the network parameters in S-LL. However, its effect may weaken over time due to the presence of distribution shift and temporal coherence in the input data-stream. Furthermore, the initialization of the prior with the old posterior at each incremental step can introduce information loss in the network for a longer sequence of S-LL. On these grounds, we propose a functional regularizer, which encourages the network to mimic the output responses as produced in the past for the previously observed samples. Specifically, we propose to minimize KL divergence between class-probability scores obtained in past and current incremental step t. In particular, we uniformly select N'_2 samples along with their logits and minimize the objective in Eq. 5, or equivalently under mild assumptions of knowledge-distillation [23], we minimize the mean-squared-error between the corresponding logits as in Eq. 6:

$$\boldsymbol{L}^2_t(\boldsymbol{\theta}) = \lambda_2 \cdot \sum_{j=1}^{N'_2} \mathbb{E}_{\boldsymbol{\theta}\sim q_t(\boldsymbol{\theta})}\left[\mathrm{KL}\left(\mathrm{smax}(\boldsymbol{h}_j)\;||\;\mathrm{smax}(f_{\boldsymbol{\theta}}(\boldsymbol{z}_j))\right)\right] \tag{5}$$

$$= \lambda_2 \cdot \sum_{j=1}^{N'_2} \mathbb{E}_{\boldsymbol{\theta}\sim q_t(\boldsymbol{\theta})}\left[\left|\left|\boldsymbol{h}_j - f_{\boldsymbol{\theta}}(\boldsymbol{z}_j)\right|\right|^2_2\right] \tag{6}$$

where: (i) $f(\cdot)$: plastic network without smax (softmax) activation, (ii) λ_2: hyper-parameter, (iii) $\boldsymbol{h}_j$: logits.

One important highlight of Eq. 6 is that the stored logits are updated in an online manner. That is, we replace the

TABLE II: Memory buffer capacity used for various datasets.

Dataset	ImageNet100	iCubWorld 1.0	CORe50
Buffer Capacity	1000	180	1000
Training-Set Size	127778	6002	119894

stored logits with the newly predicted logits whenever a sample is used for rehearsal. Therefore, it is called *snap-shot self-distillation* [37]. It essentially prevents the model from being constrained to match the sub-optimal initial logits and minimizes the information loss in the network further.

B. Informative Past Sample Selection For Replay

We consider the following strategies for selecting past informative samples for memory replay:

Uniform Sampling (Uni). In this approach, samples are selected uniformly random from memory.

Uncertainty-Aware Positive-Negative Sampling (UAPN). UAPN selects $N_1'/2$ samples with the highest uncertainty scores (negative samples) and $N_1'/2$ samples with the lowest uncertainty scores (positive samples).

Loss-Aware Positive-Negative Sampling (LAPN). LAPN selects $N_1'/2$ samples with the highest loss-values (negative-samples), and $N_1'/2$ samples with the lowest loss-values (positive-samples).

C. Memory Buffer Replacement Policy

In this work, we consider the following strategies to replace a stored example with a newly arriving sample.

Loss-Aware Weighted Class Balancing Replacement (LAWCBR). In this approach, whenever the buffer is full, we remove a sample from the majority class, i.e., $y_r = \arg\max \text{ClassCount}(\boldsymbol{M})$. We weigh each sample of the majority class inversely w.r.t their loss, i.e., $w_i^{y_r} \propto \frac{1}{l_i^{y_r}}$ and use these weights as the replacement probability; the lesser the loss, the more likely to be removed.

Loss-Aware Weighted Random Replacement With A Reservoir (LAWRRR). In this approach, we propose a novel variant of reservoir sampling [38] to replace an existing sample whenever the buffer is full. We weigh each stored sample inversely w.r.t the loss, i.e., $w_i \propto \frac{1}{l_i}$, and proportionally to the number of examples representative of that particular sample's class, i.e., $w_i \propto \text{ClassCount}(\boldsymbol{M}, y_i)$. We combine these two scores and use that as the replacement probability; the higher the weight, the more likely to be replaced.

D. Efficient Buffer Update

Loss/uncertainty-aware sampling strategies require computing these quantities in each incremental step for all the stored examples. However, it becomes computationally expensive with the larger replay buffer size. We, therefore, store the corresponding loss-values and uncertainty-scores along with the examples in the replay buffer. Since these quantities are just scalar values, the additional storage cost is negligible, but it makes the learning pipeline computationally efficient. Every time a sample is selected for memory replay, its loss and uncertainty are replaced by the new values. Additionally, we also replace the stored logits with the new logits. Empirically, we observe that the model's accuracy degrades if we don't update the logits with the new logits.

E. Training

Training the plastic network (BNN) $F(\cdot)$ requires specification of $q(\boldsymbol{\theta})$ and, in this work, we model $\boldsymbol{\theta}$ by stacking up the parameters (weights & biases) of the network $F(\cdot)$, as similar to [8]. We use a Gaussian mean-field posterior $q_t(\boldsymbol{\theta})$ for the network parameters, and choose the prior distribution, i.e., $q_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$, as multivariate Gaussian distribution. We train the network $F(\cdot)$ by maximizing the ELBO in Eq. 4 and minimizing the mean-squared-error in Eq. 6. For memory replay in Eq. 4, we select past informative samples using the strategies mentioned in Section III-B, and we use *uniform sampling* to select samples from memory to be used in Eq. 6. Fig. 2 shows the schematic diagram of the proposed model as well as the learning process.

IV. Experiments

A. Datasets And Data Orderings

Datasets. To evaluate the efficacy of the proposed model we perform extensive experiments on three benchmark datasets: ImageNet100, iCubWorld 1.0 [13], and CORe50 [39]. ImageNet100 is a subset of ImageNet-1000 (ILSVRC-2012) [31] containing randomly chosen 100 classes, with each class containing 700-1300 training samples and 50 validation samples. Since the test data for ImageNet-1000 (ILSVRC-2012) [31] is not provided with labels, we use the validation data for evaluating the model's performance, similar to [5]. iCubWorld 1.0 is an object recognition dataset which contains the sequences of video frames, with each containing a single object. There are 10 classes, each containing 3 different object instances with each class containing 600-602 samples for training and 200-201 samples for testing. CORe50 is similar to iCubWorld 1.0, containing images from temporally coherent sessions. There are 10 classes, each containing 5 different object instances with each class containing 11983-12000 samples for training and 4495-4500 samples for testing. *Technically, iCubWorld, and CORe50 are the ideal dataset for streaming learning, as it requires learning from temporally coherent image sequences, which are naturally non-i.i.d images*.

Evaluation Over Different Data Orderings. The proposed approach is robust to the various *S-LL* settings; we evaluate the model's S-LL ability with the following four [4], [5] challenging data ordering schemes: (i) *'streaming iid'*: where the data-stream is organized by the randomly shuffled samples from the dataset, (ii) *'streaming class iid'*: where the data-stream is organized by the samples from one or more classes, these samples are shuffled randomly, (iii) *'streaming instance'*: where the data-stream is organized by temporally ordered samples from different object instances, and (iv) *'streaming class instance'*: where the data-stream is organized by the samples from different classes, the samples within a class are temporally ordered based on different object instances. *Only iCubWorld 1.0 and CORe50 dataset contains the temporal ordering, therefore 'streaming instance', and 'streaming class instance' settings are evaluated only on these two datasets.*

TABLE III: $\mathbf{\Omega}_{\text{all}}$ results with their associated standard deviations. For each experiment, the method with best performance in *'streaming-learning-setup'* is highlighted in **Bold**. The reported results are averaged over 10 runs with different permutations of the data. Offline model is trained only once. $\widehat{\text{Offline}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\alpha}_{\text{offline},t}$, where T is the total number of testing events. '-' indicates experiments we are unable to run, because of compatibility issues.

Method	iid			Class-iid			instance		Class-instance	
	iCubWorld 1.0	CORe50	ImageNet100	iCubWorld 1.0	CORe50	ImageNet100	iCubWorld 1.0	CORe50	iCubWorld 1.0	CORe50
Fine-Tune	0.14 ± 0.02	0.11 ± 0.00	0.01 ± 0.00	0.39 ± 0.05	0.35 ± 0.02	0.12 ± 0.01	0.13 ± 0.00	0.11 ± 0.00	0.35 ± 0.00	0.34 ± 0.00
EWC	-	-	-	0.38 ± 0.04	0.35 ± 0.02	0.12 ± 0.00	-	-	0.35 ± 0.00	0.34 ± 0.00
MAS	-	-	-	0.39 ± 0.06	0.34 ± 0.00	0.12 ± 0.00	-	-	0.35 ± 0.00	0.34 ± 0.00
VCL	-	-	-	0.38 ± 0.05	0.35 ± 0.01	0.12 ± 0.00	-	-	0.35 ± 0.00	0.34 ± 0.00
Coreset VCL	-	-	-	0.39 ± 0.06	0.34 ± 0.00	0.13 ± 0.01	-	-	0.47 ± 0.02	0.47 ± 0.00
GDumb	0.90 ± 0.04	0.93 ± 0.01	0.84 ± 0.01	***0.97 ± 0.02***	***0.97 ± 0.01***	***0.92 ± 0.01***	0.67 ± 0.05	0.74 ± 0.02	0.79 ± 0.03	0.65 ± 0.03
DER	0.14 ± 0.04	0.11 ± 0.00	0.01 ± 0.00	0.41 ± 0.10	0.34 ± 0.00	0.12 ± 0.00	0.38 ± 0.24	0.12 ± 0.01	0.41 ± 0.17	0.33 ± 0.04
DER++	0.14 ± 0.04	0.11 ± 0.00	0.01 ± 0.00	0.45 ± 0.13	0.34 ± 0.00	0.12 ± 0.01	0.45 ± 0.25	0.11 ± 0.00	0.45 ± 0.23	0.34 ± 0.00
TinyER	0.96 ± 0.04	1.00 ± 0.01	0.94 ± 0.01	0.91 ± 0.03	0.96 ± 0.01	0.90 ± 0.01	0.87 ± 0.06	0.84 ± 0.03	0.82 ± 0.03	0.85 ± 0.02
ExStream	0.92 ± 0.06	0.98 ± 0.02	0.93 ± 0.01	0.88 ± 0.03	0.88 ± 0.02	0.88 ± 0.01	0.90 ± 0.05	0.83 ± 0.03	0.87 ± 0.02	0.88 ± 0.02
REMIND	0.93 ± 0.03	0.99 ± 0.01	0.91 ± 0.01	0.86 ± 0.03	0.94 ± 0.01	0.88 ± 0.02	0.82 ± 0.06	0.85 ± 0.02	0.76 ± 0.03	0.78 ± 0.04
Ours	**0.97 ± 0.01**	**1.01 ± 0.01**	**0.96 ± 0.01**	**0.95 ± 0.02**	**0.97 ± 0.01**	**0.92 ± 0.01**	**0.96 ± 0.03**	**0.98 ± 0.01**	**0.96 ± 0.02**	**0.94 ± 0.01**
Offline	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
$\widehat{\text{Offline}}$	0.7626	0.8733	0.8520	0.8849	0.9070	0.8953	0.7646	0.8733	0.8840	0.9079

B. Metrics

For evaluating the performance of the streaming learner, we use $\mathbf{\Omega}_{\text{all}}$ metric, similar to [40], [4], [5], where $\mathbf{\Omega}_{\text{all}}$ represents normalized incremental learning performance with respect to an offline learner: $\mathbf{\Omega}_{\text{all}} = \frac{1}{T}\sum_{t=1}^{T} \frac{\boldsymbol{\alpha}_t}{\boldsymbol{\alpha}_{\text{offline},t}}$, where T is the total number of testing events, $\boldsymbol{\alpha}_t$ is the performance of the incremental learner at time t, and $\boldsymbol{\alpha}_{\text{offline},t}$ is the performance of a traditional offline model at time t.

C. Baselines And Compared Methods

The proposed approach follows the *'streaming learning setup'*; to the best of our knowledge, recent works ExStream [4] and REMIND [5] are the only methods that follow the same setting. We compare our approach against these strong baselines. We also compare our model with (i) a network trained with one sample at a time (Fine-tuning/lower-bound) and (ii) a network trained offline, assuming all the data is available (Offline/upper-bound). Finally, we choose recent popular 'batch' (IBL) and 'online' learning methods, such as EWC [3], MAS [7], VCL [8], Coreset VCL [8], TinyER [15], GDumb [14], and DER/DER++ [16] as baselines and rigorously evaluate our model against these approaches. For a fair comparison, we train all the methods in an S-LL setup, i.e., one sample at a time, and no finetuning is used. Only *Coreset VCL* and *GDumb* finetune before each inference, therefore, these two methods utilize an extra advantage over the other baselines. However, BaSiL still outperforms these approaches by a significant margin.

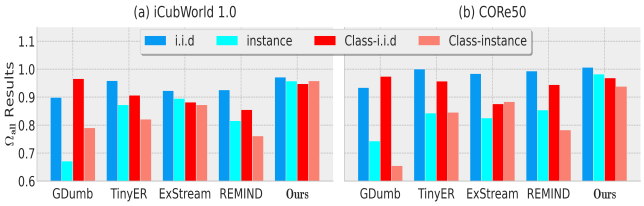

Fig. 3: Plots of $\mathbf{\Omega}_{\text{all}}$ as a function of streaming learning model and data-ordering on (a) iCubWorld 1.0, and (b) CORe50.

D. Results

The detailed results of BaSiL over various experimental settings along with the strong baseline methods are shown in Table III. We can clearly observe that BaSiL consistently outperforms all the baseline by a significant margin. The proposed model is also robust to the different streaming learning scenarios compared to the baselines. We repeat our experiment ten times and report the average accuracy. We observe that 'batch-learning' methods severely suffer from catastrophic forgetting. Moreover, replay-based 'online-learning' methods such as DER/DER++ also suffer from information loss badly.

While *GDumb* achieves higher final accuracy on class-i.i.d ordering, it finetunes before each inference. Therefore, it enjoys an extra advantage over other S-LL baselines and it is not considered as the best-performing model, even when it achieves higher accuracy.

We believe it is important to highlight that iCubWorld 1.0 and CORe50 are two challenging datasets, which evaluate the models in more realistic scenarios or data-orderings. Particularly, class-instance and instance ordering require the agent to learn from temporally ordered video frames one at a time. From Table III, we observe that BaSiL obtain up to 8.58% & 6.26% improvement on iCubWorld 1.0, and 5.47% & 12.8% improvement on CORe50, over the state-of-the-art streaming learning approaches. Fig. 3 shows the impact of temporal orderings on the streaming learning model's performance. It is evident that class-instance and instance ordering are more difficult, and the baselines continue to suffer from severe forgetting.

Finally, for completeness, we train BaSiL in 'batch' as well as 'online' learning setting to determine its effectiveness and compatibility in these settings. In Fig. 4, we compare BaSiL in various settings. We can observe that BaSiL outperforms the baselines by a significant margin on both class-i.i.d and class-instance ordering on iCubWorld, *implying even though BaSiL is designed to work in S-LL, it can be thought of as a robust method for various LL scenarios with the widest possible applicability*.

E. Implementation Details

In all the experiments, models are trained with one sample at a time. Only *Coreset VCL* and *GDumb* finetune before each inference. For a fair comparison, the same network

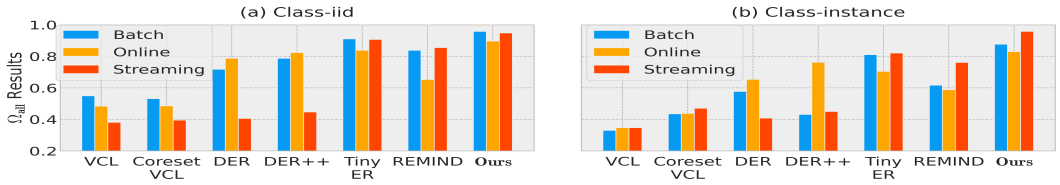

Fig. 4: $\mathbf{\Omega}_{\text{all}}$ results for different versions of baselines on iCubWorld 1.0 on (a) Class-iid, and (b) Class-instance ordering.

structure is used throughout all the models. For all methods, we use fully connected single-head networks with two hidden layers as the plastic network $F(\cdot)$, where each layer contains 256 nodes with ReLU activations; for 'VCL', 'Coreset VCL', and 'BaSiL', $F(\cdot)$ is a BNN, whereas for all other methods $F(\cdot)$ is a deterministic network. We store the same number of past examples for all replay-based approaches, as mentioned in Table II, and use same number of replay samples throughout all rehearsal-based approaches. We use Mobilenet-V2 [41] pre-trained on ImageNet-1000 (ILSVRC-2012) [31] as the visual feature extractor. For memory-replay, we use *'uncertainty-aware positive-negative'* sampling strategy throughout all data-orderings, except for 'streaming-i.i.d' ordering, we use *'uniform'* sampling. We use *'loss-aware weighted random replacement with a reservoir'* sampling strategy as memory replacement policy for all the experiments. We set $N_1' = N_2' = 16$ throughout all experiments. We set $\lambda_1 = 1$ and $\lambda_2 = 0.3$ across all experiments; however, for *online/batch* learning experiments, we use $\lambda_2 = 0.2$ and use *uniform sampling* for memory replay. We repeat our experiment 10 times and report the average-accuracy.

V. Ablation Study

We perform extensive ablation to validate the significance of the proposed components.

Significance Of Different Sampling Strategies. In Table IV, we compare the performance of BaSiL while using various sampling strategies and memory replacement policies. We observe that for the buffer replacement, LAWRRR performs better compared to LAWCBR. Furthermore, for the sample replay, UAPN, along with LAWRRR memory buffer policy, outperforms other sampling strategies, except *uniform sampling* (Uni) performs better on i.i.d ordering.

Choice Of Hyperparameter (λ_2). Fig. 5 shows the effect of changing the knowledge-distillation loss weight λ_2 on the final $\mathbf{\Omega}_{\text{all}}$ accuracy for iCubWorld 1.0 on instance and class-instance ordering, while using different sampling strategies and buffer replacement policies. We observe the best model performance for $\lambda_2 = 0.3$, and use this value for all our experiments.

Significance Of Knowledge-Distillation Loss. Fig. 5 with $\lambda_2 = 0.0$ represents the model without kd-loss. We can observe that the model performance significantly degrades without knowledge distillation. Therefore, kd-loss is a key component to the model's performance.

Choice Of Buffer Capacity. We perform an ablation for different buffer capacities. We use CIFAR-10/100 [42], and the results are shown in Fig. 6. It is evident that, with the longer sequence of incoming data, BaSiL's performance improves with the increased buffer capacity, as it helps minimize the confusion in the output prediction.

TABLE IV: $\mathbf{\Omega}_{\text{all}}$ Results as a function of different sampling and buffer management policies.

Memory Replacement	Sample Selection	iCubWrold		ImageNet100	
		instance	Class instance	iid	Class-iid
LAWCBR	Uni	0.8975	0.8506	0.9582	0.9014
	UAPN	0.9346	0.8500	0.9327	0.9135
	LAPN	0.9172	0.8536	0.9253	0.9122
LAWRRR	Uni	0.9269	0.9346	**0.9640**	0.8643
	UAPN	**0.9580**	**0.9585**	0.9578	**0.9171**
	LAPN	0.9558	0.9497	0.9575	0.9112

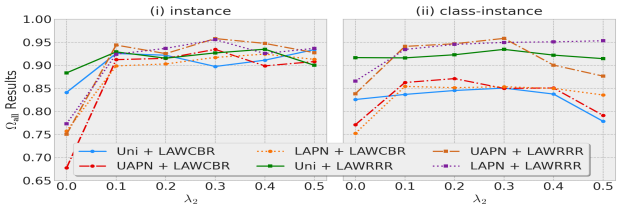

Fig. 5: Plots of $\mathbf{\Omega}_{\text{all}}$ as a function of hyper-parameter λ_2 and different sampling strategies and replacement policies for (i) instance, (ii) class-instance ordering on iCubWorld 1.0.

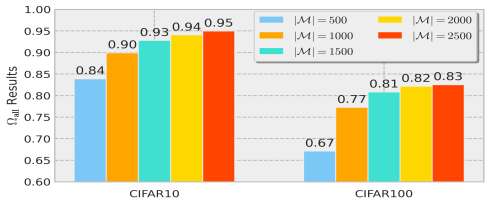

Fig. 6: Plots of $\mathbf{\Omega}_{\text{all}}$ as a function of buffer capacity $|M|$ for class-i.i.d data-ordering on CIFAR10 and CIFAR100.

VI. Conclusion

Streaming continual learning (SCL) is the most challenging and realistic framework for CL; most of the recent promising models for CL are unable to handle this above setting. Our work proposes a dual regularization and loss-aware buffer replacement to handle the SCL scenario. The proposed model is highly efficient since it learns a joint likelihood from the current and replay samples without leveraging any external finetuning. Also, to improve the training efficiency further, the proposed model selects a few most informative samples from the buffer instead of using the entire buffer for the replay. We have conducted a rigorous experiment over several challenging datasets and showed that BaSiL outperforms the recent state-of-the-art approaches in this setting by a significant margin. To disentangle the importance of the various components, we perform extensive ablation studies and observe that the proposed components are essential to handle the SCL setting.

REFERENCES

[1] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[2] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[3] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[4] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory efficient experience replay for streaming learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9769–9776.

[5] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," *arXiv preprint arXiv:1910.02509*, 2019.

[6] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 532–547.

[7] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[8] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," *arXiv preprint arXiv:1710.10628*, 2017.

[9] A. Rios and L. Itti, "Closed-loop memory gan for continual learning," *arXiv preprint arXiv:1811.01146*, 2018.

[10] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[11] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3987–3995.

[12] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[13] S. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone, "icub world: Friendly robots help building good vision data-sets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 700–705.

[14] A. Prabhu, P. H. Torr, and P. K. Dokania, "Gdumb: A simple approach that questions our progress in continual learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 524–540.

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

[16] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *arXiv preprint arXiv:2004.07211*, 2020.

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

[18] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[19] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[20] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–82.

[21] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," *arXiv preprint arXiv:1801.01423*, 2018.

[22] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[24] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in neural information processing systems*, 2017, pp. 4652–4662.

[25] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *Neural computation*, vol. 33, no. 11, pp. 2908–2950, 2021.

[26] Q. Yang, F. Feng, and R. H. Chan, "A benchmark and empirical analysis for replay strategies in continual learning," in *Continual Semi-Supervised Learning: First International Workshop, CSSL 2021, Virtual Event, August 19–20, 2021, Revised Selected Papers*. Springer, 2022, pp. 75–90.

[27] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," *arXiv preprint arXiv:1802.10269*, 2018.

[28] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[29] S. Farquhar and Y. Gal, "A unifying bayesian view of continual learning," *arXiv preprint arXiv:1902.06494*, 2019.

[30] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, *et al.*, "Memory replay gans: Learning to generate new categories without forgetting," in *Advances In Neural Information Processing Systems*, 2018, pp. 5962–5972.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[32] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[33] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

[34] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. I. Jordan, "Streaming variational bayes," *arXiv preprint arXiv:1307.6769*, 2013.

[35] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[36] S. Ghosh, J. Yao, and F. Doshi-Velez, "Structured variational learning of bayesian neural networks with horseshoe priors," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1744–1753.

[37] C. Yang, L. Xie, C. Su, and A. L. Yuille, "Snapshot distillation: Teacher-student optimization in one generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2859–2868.

[38] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[39] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 17–26. [Online]. Available: https://proceedings.mlr.press/v78/lomonaco17a.html

[40] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[42] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.