

Simple Lifelong Learning Machines

σ (Joshua T. Vogelstein, Jayanta Dey*)¹, Hayden S. Helm,¹ Will LeVine,¹ Ronak D. Mehta,¹ Tyler M. Tomita,¹ Haoyin Xu,¹ Ali Geisa,¹ Qingyang Wang,¹ Gido M. van de Ven,^{2,3} Chenyu Gao,¹ Weiwei Yang,⁴ Bryan Tower,⁴ Jonathan Larson,⁴ Christopher M. White,⁴ and Carey E. Priebe¹

Abstract—In lifelong learning, data are used to improve performance not only on the present task, but also on past and future (unencountered) tasks. While typical transfer learning algorithms can improve performance on future tasks, their performance on prior tasks degrades upon learning new tasks (called forgetting). Many recent approaches for continual or lifelong learning have attempted to *maintain* performance on old tasks given new tasks. But striving to avoid forgetting sets the goal unnecessarily low. The goal of lifelong learning should be to use data to improve performance on both future tasks (forward transfer) and past tasks (backward transfer). In this paper, we show that a simple approach—representation ensembling—demonstrates both forward and backward transfer in a variety of simulated and benchmark data scenarios, including tabular, vision (CIFAR-100, 5-dataset, Split Mini-Imagenet, Food1k, and CORE50), and speech (spoken digit), in contrast to various reference algorithms, which typically failed to transfer either forward or backward, or both. Moreover, our proposed approach can flexibly operate with or without a computational budget.

Index Terms—continual/lifelong learning, forward transfer, backward transfer, replay

I. INTRODUCTION

LEARNING is a process by which an intelligent system improves performance on a given task by leveraging data [1]. In classical machine learning, the system is often optimized for a single task [2, 3]. While it is relatively easy to *simultaneously* optimize for multiple tasks (multi-task learning) [4], it has proven much more difficult to *sequentially* optimize for multiple tasks [5, 6]. Specifically, classical machine learning systems, and natural extensions thereof, exhibit “catastrophic forgetting” when trained sequentially, meaning their performance on the prior tasks drops precipitously upon training on new tasks [7, 8, 9]. However, learning could be lifelong, with agents continually building on past knowledge and experiences, improving on many tasks given data associated with any task. For example, in humans, learning a second language often improves performance in an individual’s native language [10].

In lifelong learning, where tasks arrive sequentially, the ability to transfer knowledge across tasks is characterized by two complementary objectives: *forward transfer* and *backward transfer*. *Forward transfer* facilitates accelerated learning in new tasks using previous knowledge. In contrast, *backward transfer* evaluates the impact of new learning on previously

encountered tasks. Achieving both positive forward and backward transfer is crucial for an effective lifelong learner. However, as we will demonstrate, many existing lifelong learning algorithms do not enable forward transfer to future tasks, and most do not exhibit positive backward transfer to previously learned tasks.

In this paper, we propose a general and simple approach for lifelong learning which can be used with many existing encoder models. Specifically, we focus our approach on ensembling deep networks (Simple Lifelong Learning Networks, SILLY-N). Additionally, we demonstrate how the same approach can be generalized for lifelong learning based on ensembling decision forests (Simple Lifelong Learning Forests, SILLY-F). Table I and Figure 3 show our proposed approaches grow linearly with the task number and can flexibly operate with both growing and constant resources depending on the available computation budget. Moreover, we explore our proposed algorithm as compared to a number of reference algorithms on an extensive suite of numerical experiments that span simulation, vision datasets including CIFAR-100, 5-dataset, Split Mini-Imagenet, and Food1k, as well as the spoken digit dataset. Figure 1 illustrates that our algorithm outperforms all the reference algorithms in terms of forward, backward, and overall transfer on different vision and speech datasets. Ablation studies indicate the degree to which the amount of representation or storage capacity and replaying old task data impact performance of our algorithms. All our code and experiments are open source to facilitate reproducibility.

The subsequent organization of the paper can be summarized as:

- 1) Section II presents a discussion of relevant algorithms, highlighting their architecture and computational complexity in comparison to SILLY-N and SILLY-F.
- 2) Section III introduces the learning environment and proposes three transfer statistics.
- 3) Section IV details the design of SILLY-N and SILLY-F, incorporating insights from various ensembling approaches.
- 4) Sections V and VI provide empirical evaluations of SILLY-N and SILLY-F on both simulated and benchmark datasets.
- 5) Section VII concludes with a discussion of strengths, limitations, and directions for future research.

II. RELATED WORKS

In this work, we propose a lifelong learning approach based on representation ensembling which can flexibly operate

¹Johns Hopkins University, ²Baylor College of Medicine, ³University of Cambridge, ⁴Microsoft Research

σ denotes equal contribution (author order was decided using a coin flip and both authors have the right to list their name first in their CV).

* corresponding author: jdey4@jhmi.edu

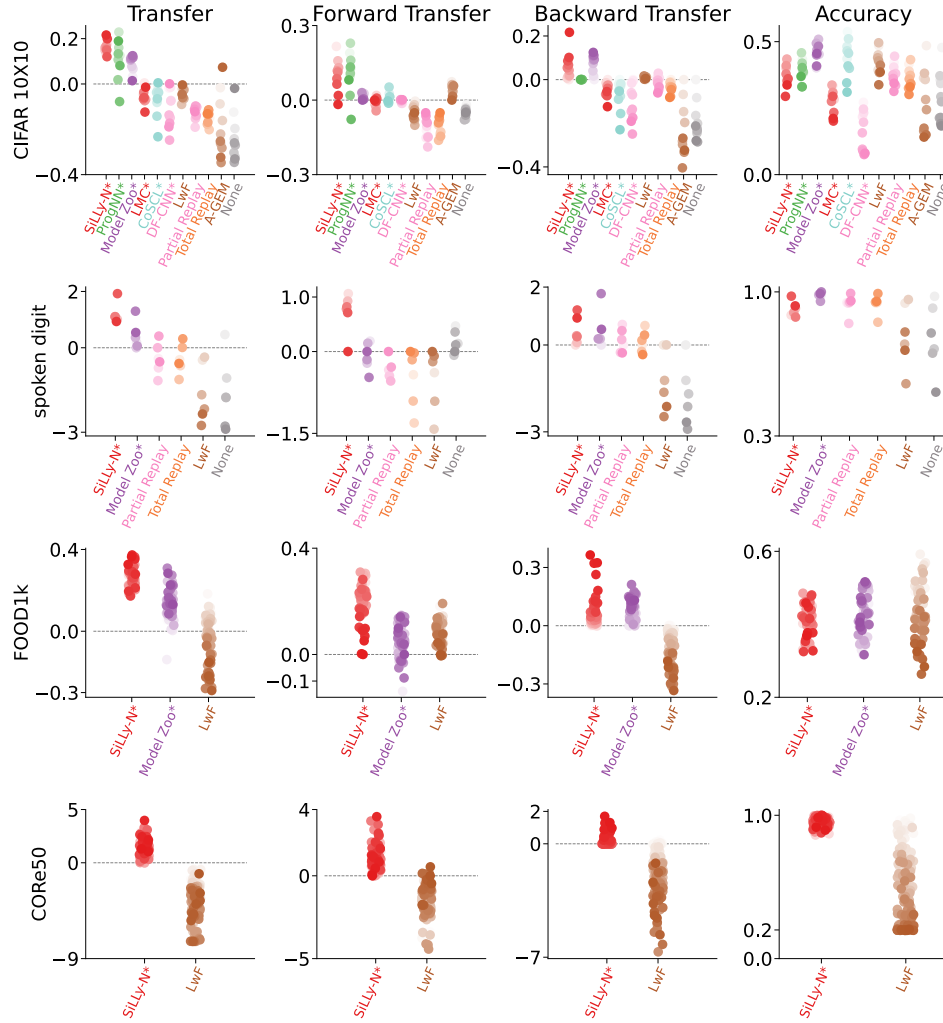


Fig. 1. **Performance summary on different vision and speech benchmark datasets.** Columns are different evaluation criteria (see Section III for definitions, and Section VI for experimental details), each strip of colored dots corresponds to an algorithm (we introduce `SILLY-N` here) and each dot represents a task. Older tasks have darker colors. Resource growing algorithms have a ‘*’. EWC, O-EWC, SI, TAG and ER always perform worse than LwF, and hence we do not show them in the plot. `SILLY-N` (red) outperforms all reference algorithms in terms of overall (first column), forward (second column), and backward (third column). Importantly, such better transfer is achieved at high overall accuracy (fourth column). For CORE50 dataset, we are unable to run `MODEL ZOO` in the resource constrained environment of the experiment and LwF completely forgets the first several tasks with the task-accuracies going down to the chance level (20%). See Appendix Figure 1 for the error bars for each dot in the figure.

in both resource growing and resource constrained modes. Existing resource constrained algorithms, such as EWC [11], ONLINE EWC [12], EWC++ [13], SI [14], and LwF [15], use regularization techniques to exploit the stability-plasticity trade-off to mitigate forgetting.

On the contrary, the resource growing approaches add resources as they face more tasks. Authors from [16] used a weighted ensemble of learners in a streaming setting with distribution shift. `TRADABOOST` [18] boosts ensemble of learners to enable transfer learning. In continual learning scenarios, many algorithms have been built on these ideas by ensembling dependent representations. For example, `LEARN++` [19] boosts ensembles of weak learners learned over different data sequences in class incremental lifelong learning settings [20]. `MODEL ZOO` [21] uses the same boosting approach in task incremental lifelong learning scenarios. Another group of algorithms, `PROGNN` [22] and `DF-CNN` [23] learn a new “column”

of nodes and edges with each new task, and ensembles the columns for inference (such approaches are commonly called ‘modular’ now). The primary difference between `PROGNN` and `DF-CNN` is that `PROGNN` has forward connections to the current column from all the past columns. This creates the possibility of forward transfer while freezing backward transfer. However, the forward connections in `PROGNN` render it computationally inefficient for a large number of tasks. `DF-CNN` gets around this problem by learning a common knowledge base and thereby, creating the possibility of backward transfer.

Recently, many other modular approaches have been proposed in the literature that improve on `PROGNN`’s capacity growth. These methods consider the capacity for each task being composed of modules that can be shared across tasks and grown as necessary. For example, `PACKNET` [24] starts with a fixed capacity network and trains for additional tasks

by freeing up portion of the network capacity using iterative pruning. Veniat et al. [25] trains additional modules with each new task, and the old modules are only used selectively. [26] improved the memory efficiency of the modular methods by adding new modules according to the complexity of the new tasks. Authors in [27] proposed non-parametric factorization of the layer weights that promotes sharing of the weights between tasks. However, all of modular methods described above lack backward transfer because the old modules are not updated with the new tasks. Dynamically Expandable Representation (DER) [28] proposed an improvement over the modular approaches where the model capacity is dynamically expanded and the model is fine-tuned by replaying a portion of the old task data along with the new task data. This approach achieves backward transfer between tasks as reported by the authors in the experiments. Another modular approach, proposed by Kang et al. [29], uses sub-networks inspired by the lottery ticket hypothesis [30] and dynamically expands the network capacity with the number of tasks.

Another strategy for building lifelong learning machines is to use total or partial replay [31, 32, 33, 34, 35]. Replay approaches keep the old data and replay them when faced with new tasks to mitigate catastrophic forgetting. Many recent approaches combine replay with other transfer learning strategies like knowledge distillation [37] to improve storage efficiency of replay data. Recently, hypernetwork based old network weight generation accompanied by synthetic replay [38] demonstrated potential of memory-efficient operation in a learning environment with a huge number of tasks. Another replay algorithm, MER [39], aligns the gradient update between task datasets while replaying old task data. However, as we illustrated in the main text, previously proposed replay algorithms do not demonstrate positive backward transfer in our experiments, though they often do not forget as much as other approaches.

Our approach builds directly on previously proposed modular and replay approaches with one key distinction: in our approach, representations are learned independently. Independent representations also have computational advantages, as doing so merely requires quasilinear time and space, and can be learned in parallel.

III. MATHEMATICAL FRAMEWORK

A. The lifelong learning objective

In a lifelong learning setting, we consider a sequence of tasks $\mathcal{T} = \{1, 2, \dots, T\}$, where the tasks are known during training and testing. Each task $t \in \mathcal{T}$ shares the same input space, $\mathcal{X} \subset \mathbb{R}^D$, with class labels $\mathcal{Y} = \{1, \dots, K_t\}$. The tasks arrive sequentially, and within each task t , the dataset $\mathbf{S}^t = \{(X_i, Y_i)\}_{i=1}^{n_t}$ is sampled *iid* from a fixed distribution \mathcal{D}_t , with $\sum_{t=1}^T n_t = n$.

Given access to all observed datasets $\bigcup_{t=1}^T \mathbf{S}^t$, the goal is to find a learner f that minimizes the overall generalization error across all tasks:

$$\begin{aligned} & \text{minimize} && \sum_{t=1}^T \mathcal{E}_f(\bigcup_{t'=1}^T \mathbf{S}^{t'}) \\ & \text{subject to} && f \in \mathcal{F} \end{aligned}, \quad (1)$$

where \mathcal{E}_f^t is the generalization error or expected risk on task t and the learner will have access to a total of T datasets after T tasks, $\bigcup_{t=1}^T \mathbf{S}^t$ ¹. Risk is defined as the expected task-specific loss $\ell_t : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty)$ in task t (see [40] for a detailed formulation on different out-of-distribution learning scenarios).

B. Lifelong learning evaluation criteria

Others have previously introduced criteria to evaluate transfer, including forward and backward transfer [25, 41, 42, 43]. Pearl [44] introduced the transfer benefit ratio, which builds directly off relative efficiency from classical statistics [45]. We define three notions of transfer building on relative efficiency.

Definition 1 (Transfer). *Overall transfer of algorithm f for a given Task t is:*

$$\text{Transfer}^t(f) := \log \frac{\mathcal{E}_f^t(\mathbf{S}^t)}{\mathcal{E}_f^t(\bigcup_{t'=1}^T \mathbf{S}^{t'})}. \quad (2)$$

In words, Equation 2 quantifies the extent that a learner f , is able to improve the performance on task t , when using the data on all other tasks, $(1, \dots, T)$. It does so by computing the ratio of generalization error on task t while only training on data from task t , relative to the generalization error on task t when training on the data from all tasks.

Forward transfer quantifies how much performance a learner transfers forward to future tasks, given prior tasks.

Definition 2 (Forward Transfer). *The forward transfer of f for task t is :*

$$\text{Forward Transfer}^t(f) := \log \frac{\mathcal{E}_f^t(\mathbf{S}^t)}{\mathcal{E}_f^t(\bigcup_{t'=1}^t \mathbf{S}^{t'})}. \quad (3)$$

Forward transfer is identical to transfer as defined in Equation 2, except that it only considers the relation between training on data from task t as compared to training on the prior tasks, $(1, \dots, t)$, excluding future tasks, $(t+1, \dots, T)$.

Backward transfer quantifies how much a learner transfers backward to previously observed tasks, in light of new tasks.

Definition 3 (Backward Transfer). *The backward transfer of f for Task t is:*

$$\text{Backward Transfer}^t(f) := \log \frac{\mathcal{E}_f^t(\bigcup_{t'=1}^t \mathbf{S}^{t'})}{\mathcal{E}_f^t(\bigcup_{t'=1}^T \mathbf{S}^{t'})}. \quad (4)$$

Backward transfer is also like Equation 2, except it compares the performance when training on all prior tasks, $(1, \dots, t)$, with training on all tasks, $(1, \dots, T)$.

Note that Transfer can be decomposed into Forward Transfer and Backward Transfer (see Appendix A for details):

$$\text{Transfer}^t(f) = \text{Forward Transfer}^t(f) + \text{Backward Transfer}^t(f). \quad (5)$$

¹More generally, we may have J datasets, where $J \neq T$ and each dataset may be associated with the target distributions of multiple tasks. For simplicity, we do not consider such scenarios further at this time.

TABLE I

A COMPUTATIONAL TAXONOMY OF LIFELONG LEARNERS. WE SHOW SOFT-O NOTATION ($\tilde{O}(\cdot, \cdot)$) AS A FUNCTION OF TOTAL TRAINING SAMPLES, $n = \sum_t^T n_t$, WHERE n_t IS THE NUMBER OF TRAINING SAMPLES FOR THE t^{th} TASK AND TOTAL TASK, T , AS WELL AS THE COMMON SETTING WHERE n IS PROPORTIONAL TO T . PARAMETRIC, SEMI-PARAMETRIC, AND NON-PARAMETRIC ALGORITHMS HAVE PARAMETERS THAT REMAIN FIXED, GROW SLOWLY, AND SCALE PROPORTIONALLY TO n , RESPECTIVELY.

Parametric	Capacity	Space		Time		Examples
	(n, T)	(n, T)	$(n \propto T)$	(n, T)	$(n \propto T)$	
parametric	1	1	1	1	1	SILLY-N-M, SILLY-F-M
parametric	1	1	1	n	n	O-EWC [12], SI [14], LwF [15]
parametric	1	T	n	nT	n^2	EWC [11]
parametric	1	n	n	nT	n^2	TOTAL REPLAY
semi-parametric	T	T^2	n^2	nT	n^2	PROGNN[22]
semi-parametric	T	T	n	n	n	DF-CNN [23]
semi-parametric	T	$T + n$	n	n	n	SILLY-N, MODEL ZOO[21], DER [28], LMC[26]
non-parametric	n	n	n	n	n	SILLY-F, IBP-WF [27]

The above equation underscores the sequential progression of tasks in lifelong learning, indicating that the total knowledge a learner acquires is derived from both previously encountered tasks (forward transfer) and those yet to be encountered (backward transfer). While forward transfer is relatively straightforward to achieve [46], backward transfer presents a significant challenge due to the issue of catastrophic forgetting. In particular, a learner that demonstrates strong forward transfer (Forward Transfer^t(f) > 0) but experiences negative backward transfer (Backward Transfer^t(f) < 0) from all future tasks will eventually lose the learned knowledge, resulting in Transfer^t(f) ≤ 0 for the task.

Another paper [25], concomitantly introduced transfer and forgetting (backward transfer). Their statistics are the same as ours, except they do not use a log. We opted for a log to address numerical stability issues in comparing small numbers. Because log is a monotonic function, the order of ranking algorithms is preserved (Appendix Figure 2 shows a version of Figure 1, but using Veniat’s statistics, which is nearly visually identical). By virtue of introducing Forward Transfer here, we can identify the inherent trade-off between forward and backward transfer, for a fixed amount of total transfer. Apart from the above statistics, we also report accuracy per task.

Definition 4 (Accuracy). *The accuracy of algorithm f on task t after observing total T datasets is:*

$$\text{Accuracy}^t(f) := 1 - \mathcal{E}_f^t\left(\bigcup_{t'=1}^T \mathbf{S}^{t'}\right). \quad (6)$$

C. Computational Taxonomy of Lifelong Learners

In the past 30 years, a number of lifelong learning algorithms have attempted to overcome catastrophic forgetting [47, 48]. These algorithms can be broadly classified into three categories: parametric, semi-parametric, and non-parametric approaches, based on their representational capacity and how it scales with the number of tasks and data samples. Table I shows a computational taxonomy of lifelong learners as a function of the sample size n and the total task T , as well as the common scenario where the sample size is fixed per task and therefore proportional to the number of tasks, $n \propto T$. For time complexity, we only consider the training

time complexity. The space complexity of the learner refers to the amount of memory space needed to store the learner [49]. We also study the representation capacity of these algorithms. Capacity is defined as the size of the subset of hypotheses that is achievable by the learning algorithm [50]. For a detailed discussion and assumptions behind the complexity analysis, see Appendix D.

IV. REPRESENTATION ENSEMBLING ALGORITHMS

Shannon proposed that a learned hypothesis can be decomposed into three components: an encoder, a channel, and a decoder [51, 52]: $h(\cdot) = w \circ v \circ u(\cdot)$. Figure 2 shows these three components as the building blocks of different learning schemas. The encoder, $u : \mathcal{X} \mapsto \tilde{\mathcal{X}}$, maps an \mathcal{X} -valued input into an internal representation space $\tilde{\mathcal{X}}$ [53, 54]. The channel $v : \tilde{\mathcal{X}} \mapsto \Delta_{\mathcal{Y}}$ maps the transformed data into a posterior distribution (or, more generally, a score). Finally, a decoder $w : \Delta_{\mathcal{Y}} \mapsto \mathcal{Y}$, produces a predicted label.

A canonical example of a single learner depicted in Figure 2A is a decision tree. Importantly, one can subsample the training data to learn different components of the tree [55, 56, 57]. For example, one can use a portion of data to learn the tree structure (which is the encoder). Then, by pushing the remaining data (sometimes called the ‘out-of-bag’ data) through the tree, one can learn posteriors in each leaf node (which are the channel). The channel thus gives scores for each data point denoting the probability of that data point belonging to a specific class. Using separate sets of data to learn the encoder and the channel results in less bias in the estimated posterior in the channels as in ‘honest trees’ [55, 56, 57]. Finally, the decoder provides the predicted class label using $\arg \max$ over the posteriors from the channel.

One can generalize the above decomposition by allowing for multiple encoders, as shown in Figure 2B. Given B different encoders, one can attach a single channel to each encoder, yielding B different channels. Doing so requires generalizing the definition of a decoder so that it would operate on multiple channels. Such a decoder ensembles the *decisions*, because here each channel provides the final output based on the encoder. This is the learning paradigm behind bagging [58] and boosting [59]; indeed, decision forests are a canonical

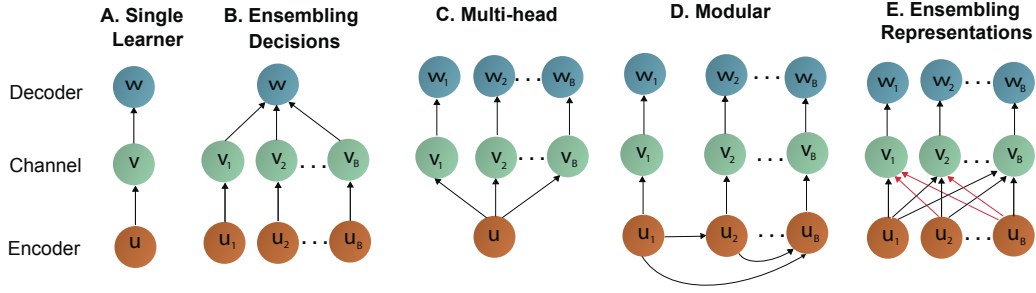


Fig. 2. Schemas of composable hypotheses. A. Single task learner. B. Ensembling decisions (as output by the channels) is a well-established practice, including random forests and gradient boosted trees. C. Learning a joint representation or D. learning future representations depending on the past encoders was previously used in lifelong learning scenarios, but encoders were not trained independently as in E. Note that the new encoders in E interact with the previous encoders through the channel layer (indicated by red arrows), thereby, enabling backward transfer. Again the old encoders interact with the future encoders (indicated by black arrows), thereby, enabling forward transfer.

example of a decision function operating on an ensemble of B outputs [60].

Although the task specific structure in Figure 2B can provide useful decision on the corresponding task, they cannot, in general, provide meaningful decisions on other tasks, because those tasks might have completely different class labels. Therefore, in the multi-head structure (Figure 2C) a single encoder is used to learn a joint representation from all the tasks, and a separate channel is learned for each task to get the score or class conditional posteriors for each task, which is followed by each task specific decoder [11, 12, 14].

Modular approaches, such as PROGNN and LMC (Figure 2D), have both multiple encoders and decoders. Connections from past to future encoders enables forward transfer. However, they freeze backward transfer.

Our approach also uses multiple encoders and decoders (Figure 2E). Unlike modular approaches, we allow interaction among encoders through the channels, including both forward and backward interactions. The result is that the channels **ensemble representations** (learned by the encoders), rather than decisions (learned by the channels as in Figure 2B). In our algorithms, we push all the data through each encoder, and each channel learns and ensembles across all encoders. When each encoder has learned complementary representations, the channels can leverage that information to improve over single task performance. This approach has applications in few-shot and multiple task scenarios, as well as lifelong learning.

A. Our representation ensembling algorithms

Figure 2E shows a general structure of our algorithm. As data from a new task arrives, the algorithm first builds a new encoder. Then, it builds the channel for this new task by pushing the new task data through *all* existing encoders. Thus the channel integrates information across all existing encoders using the new task data, thereby enabling forward transfer. At the same time, if it stores old task data (or can generate such data), it can push that data through the new encoders to update the channels from the old tasks, thereby enabling backward transfer. In either case, new test data are passed through all existing encoders and corresponding channels to make a prediction (see appendix for detailed description of this approach).

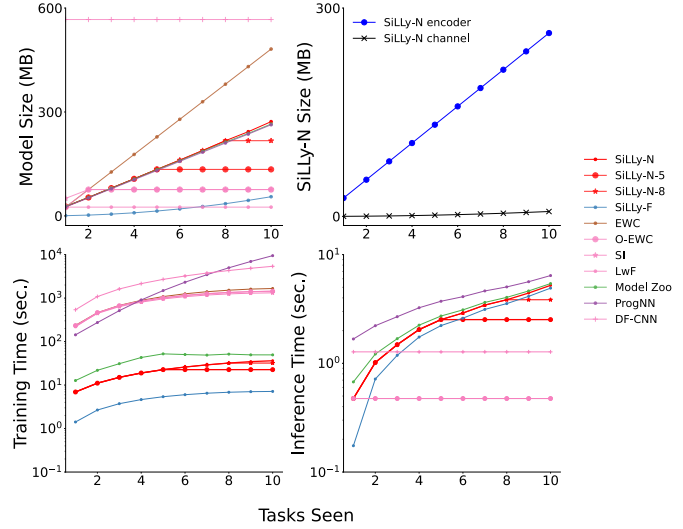


Fig. 3. **Space and time complexity as a function on number of tasks in CIFAR 10X10.** Section VI describes the experimental setup and architecture of the convolutional network used. *Top left:* Model size, SILLy can flexibly operate between resource constrained and growing modes. *Top right:* Memory consumed by SILLy-N is dominated by the encoder size. *Bottom left:* SILLy takes less training time compared to other baselines. *Bottom right:* Inference time taken by different lifelong learners for 1000 samples. Bottom row has log y-axis.

1) *Simple Lifelong Learning Networks in resource growing mode:* A Simple Lifelong Learning Network (SILLy-N) ensembles deep networks. For each task, the encoder u_t in SILLy-N is the “backbone” of a deep network (DN). Thus, each u_t maps an element of \mathcal{X} to an element of \mathbb{R}^d , where d is the number of neurons in the penultimate layer of DN. The channels are learned by averaging the outputs from decision forests [60, 61] trained on the d dimensional representations of \mathcal{X} . See Appendix Figure 3 where we conduct experiments on a CIFAR 10X10 (described later in Section VI-B1) while varying the number of decision trees per channel. Appendix Figure 3 shows that decision forest-based channels are robust to hyperparameter perturbation. Other algorithms could also be used to learn the channels, though we do not pursue them here. The decoder w_t outputs the $\arg \max$ to produce a single prediction.

2) *Simple Lifelong Learning Networks in resource constrained mode*: The above resource growing approach is ideal when the upcoming tasks become more and more complex and there is no constraint imposed by the computation and storage budget available. However, real-world scenarios often impose computational constraints. In the constant resource mode, we stop building new encoders after we have reached the computation and the storage budget imposed by the user. As new tasks arrive, we only learn new channels associated with new tasks using the old encoders. Note that this approach completely excludes the need to save old task data after we have reached the budget.

Hereafter, we will use suffix ‘-M’ after the algorithm name whenever we use resource constrained operation of SILLY-N. Here M is the total number of encoder allowed by the budget.

3) *Additional realization of our approach using random forest as encoder*: Simple Lifelong Learning Forest (SILLY-F) ensembles decision trees or forests. For each task, the encoder u_t of SILLY-F is the representation learned by a decision forest. The channel then learns the class-conditional posteriors by populating the forest leaves with out-of-task samples, as in “honest trees” [55, 56, 57]. Each channel outputs the posteriors averaged across the collection of forests learned over different tasks. The decoder w_t outputs the argmax to produce a single prediction.

Note that the amount of additional representation capacity added per task by SILLY-F is a function of the amount and complexity of the data for a new task. Contrast this with SILLY-N and other deep net based modular or representation ensembling approaches, which *a priori* choose how much additional representation to add, prior to seeing all the new task data. So, SILLY-F has capacity, space complexity, and time complexity scale with the complexity and sample size of each task. In contrast, PROGNN, SILLY-N (and others like it) have a fixed capacity for each task, even if the tasks have very different sample sizes and complexities.

Figure 3 top left shows the model size for our proposed approach grows linearly with the number of tasks. Moreover, the memory consumed by the new channels is negligible compared to the memory required to store the encoders (Figure 3 top right). The time required for inference on 1000 testing points (Figure 3 bottom right) is an order of magnitude lower compared to the time required to train a new encoder with 500 samples (Figure 3 bottom left).

V. SIMULATION DATA STUDY

A. Forward and backward transfer in a simple environment

Consider a very simple two-task environment: Gaussian XOR and Gaussian Exclusive NOR (XNOR) (Figure 4A, see Appendix E for details). The two tasks share the exact same discriminant boundaries: the coordinate axes. Thus, transferring from one task to the other merely requires learning a bit flip of the class labels. We sample a total 750 samples from XOR, followed by another 750 samples from XNOR.

SILLY-N and deep network (DN) achieve the same generalization error on XOR when training with XOR data (Figure 4Bi). But because DN does not account for a change

in task, when XNOR data appear, DN performance on XOR deteriorates (it catastrophically forgets). In contrast, SILLY-N continues to improve on XOR given XNOR data, demonstrating backward transfer. Now consider the generalization error on XNOR (Figure 4Bii). Both SILLY-N and DN are at chance levels for XNOR when only XOR data are available. When XNOR data are available, DN must unlearn everything it learned from the XOR data, and thus its performance on XNOR starts out nearly maximally inaccurate, and quickly improves. On the other hand, because SILLY-N can leverage the encoder learned using the XOR data, upon getting *any* XNOR data, it immediately performs quite well, and then continues to improve with further XNOR data, demonstrating forward transfer (Figure 4Biii). SILLY-N demonstrates positive forward and backward transfer for all sample sizes, whereas DN fails to demonstrate neither forward nor backward transfer, and eventually catastrophically forgets the previous tasks.

B. Forward and backward transfer for adversarial tasks

In the context of lifelong learning, we informally define a task t to be adversarial with respect to task t' if the true joint distribution of task t , without any domain adaptation, impedes performance on task t' . In other words, training data from task t can only add noise, rather than signal, for task t' . An adversarial task for Gaussian XOR is Gaussian XOR rotated by 45° (R-XOR) (Figure 4Aiii). Training on R-XOR therefore impedes the performance of SILLY-N on XOR, and thus backward transfer becomes negative, demonstrating graceful forgetting [62] (Figure 4Ci).

To further investigate this relationship, we design a suite of R-XOR examples, generalizing R-XOR from only 45° to any rotation angle between 0° and 90° , sampling 100 points from XOR, and another 100 from each R-XOR (Figure 4Cii). Note that we could not run the experiment for a lot of Monte Carlo repetition to have a smooth curve and hence we have shown a regressed curve fitted to the low repetition noisy curve. As the angle increases from 0° to 45° , Backward Transfer gradually decreases for SILLY-N. The 45° -XOR is the maximally adversarial R-XOR. Thus, as the angle further increases, Backward Transfer increases back up to ≈ 0.18 at 90° , which has an identical discriminant boundary to XOR. Moreover, when θ is fixed at 25° , Backward Transfer increases at different rates for different sample sizes of the source task (Figure 4Ciii).

Together, these experiments indicate that the amount of transfer can be a complicated function of (i) the difficulty of learning good representations for each task, (ii) the relationship between the two tasks, and (iii) the sample size of each.

VI. BENCHMARK DATA STUDY

For benchmark data, we build SILLY-N encoders using the network architecture described in [31]. We use the same network architecture for all benchmarking models. For the following experiments, we consider two modalities of real data: vision and language.

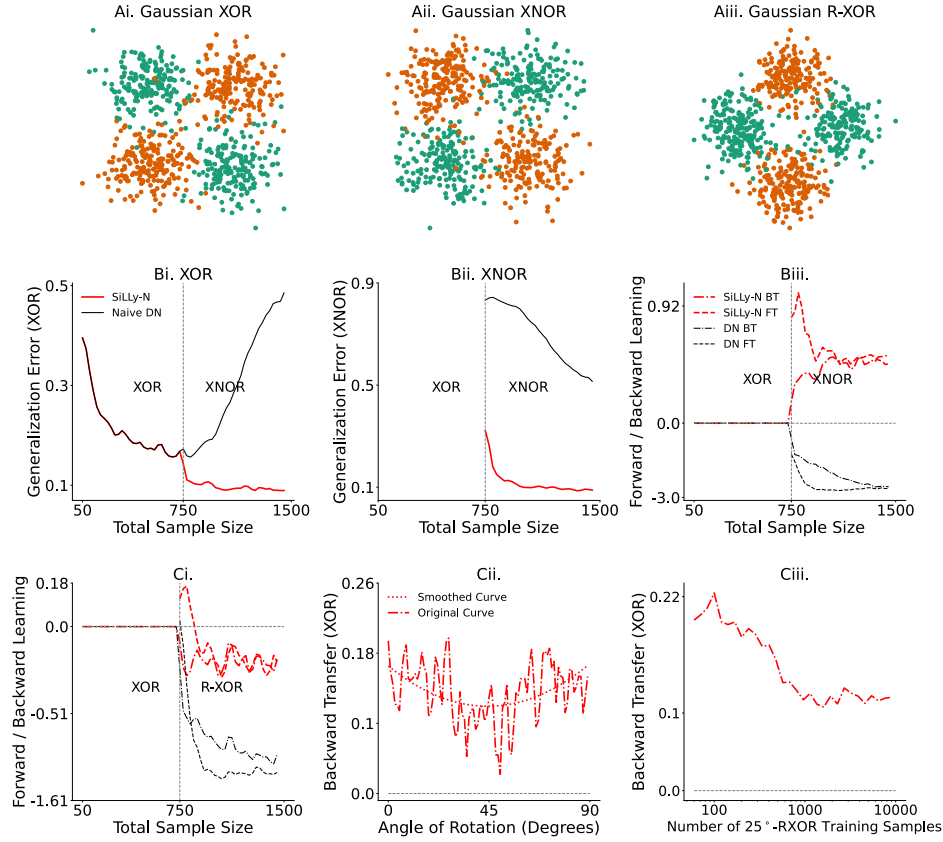


Fig. 4. **SILLY-N demonstrates forward and backward transfer.** (A) 750 samples from: (Ai) Gaussian XOR, (Aii) XNOR, which has the same optimal discriminant boundary as XOR, and (Aiii) R-XOR, which has a discriminant boundary that is uninformative, and therefore adversarial, to XOR. (Bi) Generalization error for XOR, and (Bii) XNOR. **SILLY-N** outperforms DN on XOR when XNOR data is available, and on XNOR when XOR data are available. (Biii) Forward and backward transfer of **SILLY-N** are positive for all sample sizes. (Ci) In an adversarial task setting, **SILLY-N** gracefully forgets XOR, whereas DN catastrophically forget and interfere. (Cii) Backward Transfer is maximum positive with respect to XOR when the optimal decision boundary of θ -XOR is similar to that of XOR (e.g. angles far from 45°), and negative otherwise. The dashed line shows the regression line fitted through the original points. (Ciii) Backward Transfer is a nonlinear function of the source training sample size (XOR sample size is fixed at 500).

A. Reference algorithms

We compared our approaches to 16 reference lifelong learning methods. Among them five are resource growing as well as modular approach: **PROGNN** [22], **DF-CNN** [23], **LMC** [26], **MODEL ZOO** [21], **COSCL** [63]. Note that “**MODEL ZOO**” was published after our work was archived on arXiv, and the authors have built on our work (personal communications). Other reference algorithms are resource constrained: Elastic Weight Consolidation (**EWC**) [11], Online-EWC (**O-EWC**) [12], Synaptic Intelligence (**SI**) [14], Learning without Forgetting (**LWF**) [15], **RANDUMB** [64] and “None”.

We also compare two variants of exact replay (Total Replay and Partial Replay) using the code provided in [31]. Both Total and Partial Replay store all the data they have ever seen, but Total Replay replays all of it upon acquiring a new task, whereas Partial Replay replays N samples, randomly sampled from the entire corpus, whenever we acquire a new task with N samples. The above two replay approaches can be considered as two variants of **GDUMB** [65]. Additionally, we have compared our approach with more constrained ways of replaying old task data, including Averaged Gradient Episodic Memory (**A-GEM**) [66], Experience Replay (**ER**) [34] and

Task-based Accumulated Gradients (**TAG**) [67].

For the baseline “None”, the network was incrementally trained on all tasks in the standard way while always only using the data from the current task. The implementations for all of the algorithms are adapted from open source codes [23, 68]; for implementation details, see Appendix C.

B. Core benchmarks

1) **CIFAR 10X10**: The CIFAR 100 challenge [69], consists of 50,000 training and 10,000 test samples, each a 32×32 RGB image of a common object, from one of 100 possible classes, such as apples and bicycles. CIFAR 10x10 divides these data into 10 tasks, each with 10 classes [23] (see Appendix F for details).

SILLY-N and **MODEL ZOO** demonstrate positive forward and backward transfer for every task in CIFAR 10x10, in contrast, other algorithms do not exhibit any positive backward transfer (Figure 1 first column). Moreover, they retained their accuracy while improving transfer (Figure 1 first column, first and fourth rows). **PROGNN** had a similar degree of forward transfer, but zero backward transfer, and requires quadratic

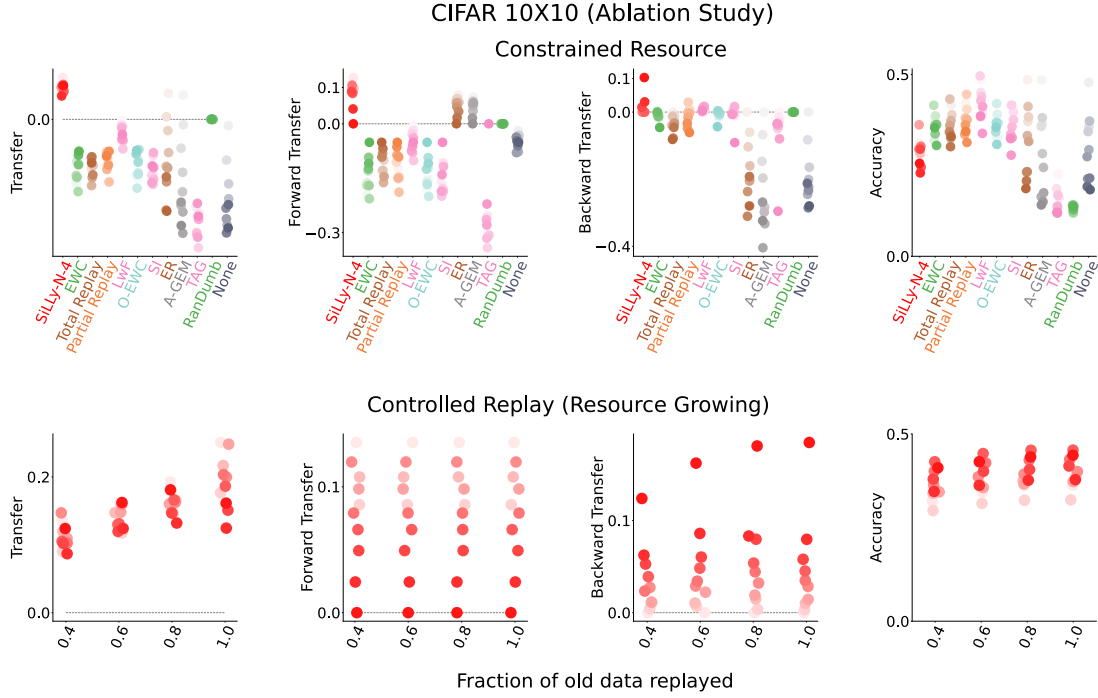


Fig. 5. **Ablation experiments on SILLY-N using CIFAR 10X10.** *Top row:* SILLY-N uniquely shows both positive forward and backward transfer while operating with the same number of parameters as other baseline constant parameter approaches. *Bottom row:* Replaying old task data impacts backward transfer while keeping forward transfer unchanged for SILLY-N.

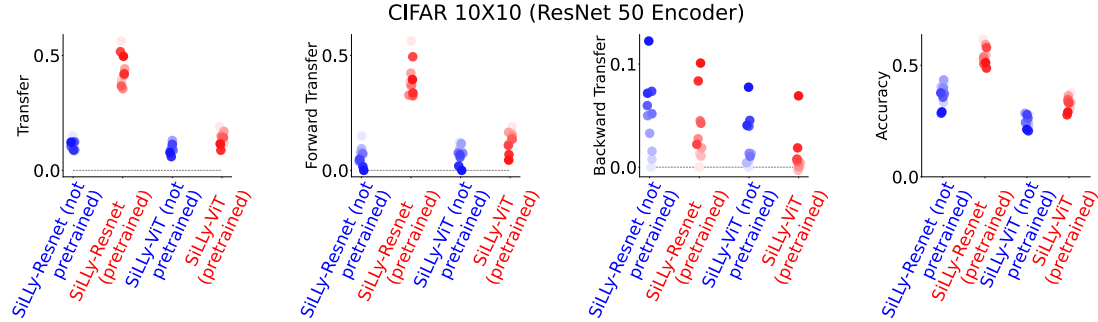


Fig. 6. **Pretrained encoders on CIFAR 10X10.** Using pretrained encoders results in better forward transfer and accuracy for SILLY-N.

space and time in sample size, unlike SILLY-N which requires quasilinear space and time.

2) *Spoken Digit:* In this experiment, we used the **Spoken Digit** dataset [70]. As shown in Figure 1 second column, SILLY-N shows positive backward and forward transfer between the spoken digit tasks, in contrast to other methods, some of which show only forward transfer, others show only backward transfer, with none showing both and some showing neither. See Appendix F for details of the experiment.

3) *FOOD1k 50X20 Dataset:* In this experiment, we use **Food1k** which is a large scale vision dataset consisting of 1000 food categories from Food2k [71]. FOOD1k 50X20 splits these data into 50 tasks with 20 classes each. For each class, we randomly sampled 60 samples per class for training the models and used rest of the data for testing purpose. Because on the CIFAR experiments MODEL ZOO performs the best among the reference resource growing models, and LWF is the best performing resource constrained algorithm, we only use

them as the reference models for the large scale experiment to avoid heavy computational cost. As shown in Figure 1 third column, SILLY-N performs the best among all the algorithms on this large dataset.

See Appendix F for experiments with datasets having more samples per task. In lifelong learning, we are often primarily concerned with situations in which we have a small number of samples per task. If we have enough samples per task, the learning agent does not need to transfer knowledge from other tasks. However, below we also experiment with non-trivial lifelong learning setting where sample per task is high.

These experiments indicate that SILLY-N has positive transfer in learning environments with various classes and sample sizes.

C. Ablation experiments

Our proposed algorithms can improve performance on all the tasks (past and future) by both growing additional re-

sources and replaying data from the past tasks. Below we do two ablation experiments using CIFAR 10X10 to measure the relative contribution of resource growth and replay to the performance of our proposed algorithms.

a) Constrained resource experiment: In this experiment, we ablate the capability of SILLY-N to grow additional resources after learning 4 encoders. We also reduce the number of channels and nodes at each encoder layer by four times to keep the total number of parameters similar to the other constant-resource-algorithms. As shown in the top row of Figure 5, SILLY-N-4 still shows positive forward and backward transfer with constant resources. However, the accuracy for SILLY-N-4 gets reduced compared to that of resource growing SILLY-N in Figure 1. Note that all the baseline algorithms have negative backward transfer. This experiment indicates that constant resource mode operation for SILLY-N may be advantageous when we have a lot of tasks to learn and have a decent amount of storage budget available. We will elaborate the above point later with a large scale dataset (food1k).

b) Controlled replay experiment: In this experiment, we train four different versions of SILLY-N sequentially on the 10 tasks from CIFAR 10X10. The only difference between different versions of the algorithms is the amount of old task data replayed. In four different versions of each algorithm, we replay 40%, 60%, 80% and 100% of the old task data respectively. As apparent from Figure 5 bottom, replaying old task data has no effect on forward transfer, but replaying more data improves backward transfer as the number of tasks increases.

These experiments indicate that (i) constraining the resource growth results in lower accuracy while still achieves positive forward and backward transfer for the algorithm, (ii) lowering the amount of replay lowers backward transfer without any effect on forward transfer.

c) Experiment using pretrained encoders: We explore the effect of using pretrained encoders on the performance of SILLY-N. For this experiment only, we use RESNET 50 and vision transformer VIT_B16 (provided in keras-vit package) [72]. We freeze all the layers excluding the final two linear layers during training. *Pretraining the encoders results in better accuracy and forward transfer, but less backward transfer for SILLY-N (Figure 6).* Vision transformers achieved lower accuracy, as expected with the small samples sizes used here [73]. Pretrained network requires less training epochs for each task (with early stopping). In this experiment, one forward pass through VIT takes ~ 5 seconds and training a VIT encoder (20 epochs) takes ~ 4 minutes using an Apple M1 Max chip and 64 GB of RAM.

D. Adversarial analysis

Consider the same CIFAR 10x10 experiment setup in Section VI-B1. In the following experiments, we modify the above setup in different adversarial settings.

a) Shuffled task order experiment: We repeat the same experiment mentioned above 5 times more by permuting the task order. Figure 7 left column shows the mean as well as the

spread of Transfer over all tasks remain the same for different shuffled task orders. Note that the encoders and thus all the channels remain the same regardless of the above task order permutation after all the tasks have been introduced, hence the distribution of Transfer for all the tasks after all the tasks have been introduced remain similar.

b) Label shuffle experiment: In this experiment, for Task 2 through 10, randomly permute the class labels within each task, rendering each of those tasks adversarial with respect to the first task (because the labels are uninformative). Figure 7 middle column indicates that SILLY-N show positive backward transfer even with such label shuffling (the other algorithms, except MODEL ZOO, did not demonstrate positive backward transfer).

c) Rotation experiment: Consider a Rotated CIFAR experiment, which uses only data from the first task, divided into two subsets of equal size (making two tasks), where the second subset is rotated by different amounts (Figure 7 right column). Backward transfer of SILLY-N is nearly invariant to rotation angle, whereas the other approaches are much more sensitive to the rotation angle. Note that the zero rotation angle corresponds to the two tasks *having identical distributions*. The fact that other algorithms fail to transfer even in this setting suggests that they may never be able to achieve a positive backward transfer. See Appendix F for an additional experiment using CIFAR 10X10.

These adversarial experiments indicate that SILLY-N is robust to adversarial perturbations in the source tasks, while most of the other algorithms are not.

E. Constant Resource Mode Operation

1) FOOD1k 50X20: The binary distinction we made above, algorithms either build resources or reallocate them, is a false dichotomy, and biologically unnatural. In biological learning, systems develop from building to fixed resources, as they grow from juveniles to adults. To explore this continuum of amount of resources to grow, we experiment on FOOD1k 50X20 dataset using the constant resource mode operation of SILLY-N as described in Section IV. We evaluate the performance of SILLY-N for different number of encoder budget. Performance of SILLY-N saturates after 30 encoders, though with only 5 encoders, still demonstrates forward and backward transfer (Figure 8).

2) CORE50 110X5: To further evaluate the effectiveness of constant resource mode operation, we utilize a large-scale dataset: CORE50 [74], partitioned into 110 tasks with 5 classes each and 100 training samples per class. To simulate a resource-constrained environment, we conducted all experiments on an Apple M1 Max chip with 64 GB of RAM. We set a budget of 20 encoders, as this configuration maintains high accuracy (above 90%) across all tasks. The fourth row of Figure 1 shows that SILLY-N achieves higher Transfer operating in constant resource mode on a large dataset, that is, CORE50 compared to the smaller datasets discussed above. In particular, BackwardTransfer drops to zero after 20 tasks due to the encoder limit, leaving only ForwardTransfer for subsequent tasks. Moreover, we cannot run resource-growing

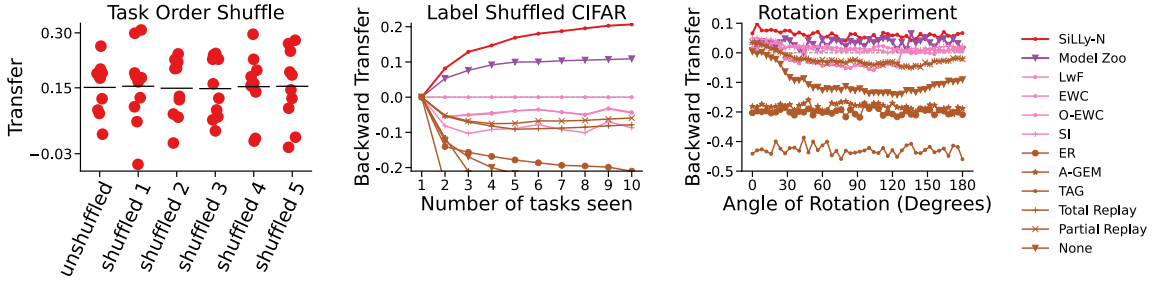


Fig. 7. **Adversarial CIFAR 10x10 experiments.** *Left:* Permuting the task sequence does not affect Transfer for SiLly-N. The mean position, indicated by the black dash, and the spread of the dots on both sides of the mean remain nearly the same. *Middle:* Shuffling class labels within tasks two through nine with 500 samples each demonstrates SiLly-N can still achieve positive backward transfer, and that the other algorithms still fail to transfer. *Right:* SiLly-N is nearly invariant to rotations, whereas other approaches are more sensitive to rotation.

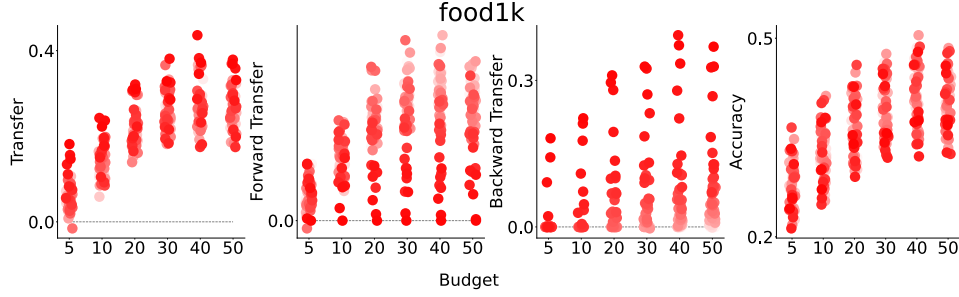


Fig. 8. **SiLly-N in constant resource mode operation.** Improvement in Transfer between tasks becomes negligible at nearly 30 encoders. The user can choose to operate in lower budget with less transfer.

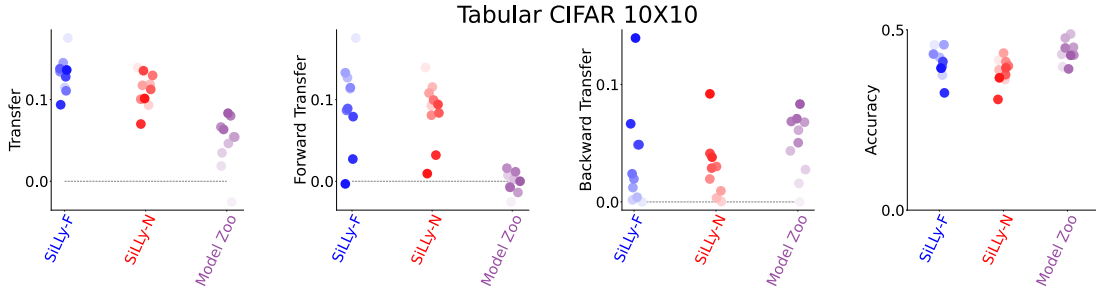


Fig. 9. **Our proposed approach can be used with random forest as encoders on tabular data (SiLly-F).** SiLly-F shows more positive forward and backward transfer while operating with less parameters compared to other baseline approaches on tabular data.

algorithms like MODEL ZOO in this resource-constrained environment. The constant resource algorithm, such as LwF, completely forgets the first several tasks, and the corresponding task accuracies go as low as the chance level, 20%.

F. SiLly-F on tabular data

In this experiment, we experiment with SiLly-F, an additional realization of our approach using random forests as encoders (described in Section IV). We flatten the CIFAR 10x10 data and use them as tabular data. We train two other best performing baseline algorithms, SiLly-N and MODEL ZOO and use three fully connected hidden layers, each having 2000 nodes, as encoders. As shown in Figure 9, SiLly-F performs the best among all the approaches. *This experiment shows our approach can be used as a general structure to do lifelong learning using other machine learning models as encoder.*

VII. DISCUSSION

We introduced representation ensembling as a simple approach for lifelong learning. Two specific algorithms, SiLly-N and SiLly-F, achieve both forward and backward transfer, by leveraging resources learned for other tasks without undue computational burdens. Our work is well suited for federated learning scenarios, where each data center independently trains a model on its private data and shares only the encoders with other centers [76]. In this paper, we have mainly focused on task-aware setting, because it is simpler. Future work will extend our approach to more challenging task-unaware settings [75]. Our code, including the code for reproducing the experiments in this manuscript, is available from <http://proglearn.neurodata.io/>.

ACKNOWLEDGMENTS

The authors thank the support of the NSF-Simons Research Collaborations on the Mathematical and Scientific Foundations

of Deep Learning (NSF grant 2031985). We also thank Raman Arora, Dinesh Jayaraman, Rene Vidal, Jeremias Sulam, Guillermo Sapiro, Michael Powell, and Weiwei Yang for helpful discussions. This work is graciously supported by the Defense Advanced Research Projects Agency (DARPA) Lifelong Learning Machines program through contracts FA8650-18-2-7834 and HR0011-18-2-0025. Research was partially supported by funding from Microsoft Research and the Kavli Neuroscience Discovery Institute.

CO-FIRST AUTHOR CONTRIBUTIONS

J.D.: Conceptualization, Methodology, Software, Data Curation, Formal Analysis, Experimentation, Theoretical proof (see Appendix D), Writing & Editing; J.T.V.: Conceptualization, Methodology, Supervision, Writing—Review & Editing, and Funding Acquisition.

REFERENCES

- [1] T. M. Mitchell, “Machine learning and data mining,” *Communications of the ACM*, vol. 42, no. 11, pp. 30–36, 1999.
- [2] V. Vapnik and A. Chervonenkis, “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities,” *Theory Probab. Appl.*, vol. 16, no. 2, pp. 264–280, Jan. 1971.
- [3] L. G. Valiant, “A Theory of the Learnable,” *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, Nov. 1984. [Online]. Available: <http://doi.acm.org/10.1145/1968.1972>
- [4] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
- [5] S. Thrun, “Is learning the n -th thing any easier than learning the first?” in *Advances in neural information processing systems*, 1996, pp. 640–646.
- [6] S. Thrun and L. Pratt, *Learning to Learn*. Springer Science & Business Media, Dec. 2012. [Online]. Available: https://market.android.com/details?id=book-X_jpBwAAQBAJ
- [7] 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.
- [8] J. L. McClelland, B. L. McNaughton, and R. C. O’Reilly, “Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory,” *Psychological review*, vol. 102, no. 3, p. 419, 1995.
- [9] T. Doan, M. A. Bennani, B. Mazouze, G. Rabusseau, and P. Alquier, “A theoretical analysis of catastrophic forgetting through the ntk overlap matrix,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1072–1080.
- [10] J. Zhao, B. Quiroz, L. Q. Dixon, and R. M. Joshi, “Comparing Bilingual to Monolingual Learners on English Spelling: A Meta-analytic Review,” *Dyslexia*, vol. 22, no. 3, pp. 193–213, Aug. 2016.
- [11] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [12] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, “Progress & compress: A scalable framework for continual learning,” *arXiv preprint arXiv:1805.06370*, 2018.
- [13] 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.
- [14] 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.
- [15] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.
- [16] H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.
- [17] I. E. Marouf, S. Roy, E. Tartaglione, and S. Lathuilière, “Weighted ensemble models are strong continual learners,” in *European Conference on Computer Vision*. Springer, 2024, pp. 306–324.
- [18] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, “Boosting for transfer learning.(2007), 193–200,” in *Proceedings of the 24th international conference on Machine learning*, 2007.
- [19] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, “Learn++: An incremental learning algorithm for supervised neural networks,” *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.
- [20] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, “Three types of incremental learning,” *Nature Machine Intelligence*, pp. 1–13, 2022.
- [21] R. Ramesh and P. Chaudhari, “Model zoo: A growing brain that learns continually,” in *International Conference on Learning Representations*, 2021.
- [22] 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.
- [23] S. Lee, J. Stokes, and E. Eaton, “Learning shared knowledge for deep lifelong learning using deconvolutional networks,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 2837–2844.
- [24] A. Mallya and S. Lazebnik, “Packnet: Adding multiple tasks to a single network by iterative pruning,” in *Pro-*

- ceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.
- [25] T. Veniat, L. Denoyer, and M. Ranzato, “Efficient continual learning with modular networks and task-driven priors,” *arXiv preprint arXiv:2012.12631*, 2020.
- [26] O. Ostapenko, P. Rodriguez, M. Caccia, and L. Charlin, “Continual learning via local module composition,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 30 298–30 312, 2021.
- [27] N. Mehta, K. Liang, V. K. Verma, and L. Carin, “Continual learning using a bayesian nonparametric dictionary of weight factors,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 100–108.
- [28] S. Yan, J. Xie, and X. He, “Der: Dynamically expandable representation for class incremental learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3014–3023.
- [29] H. Kang, R. J. L. Mina, S. R. H. Madjid, J. Yoon, M. Hasegawa-Johnson, S. J. Hwang, and C. D. Yoo, “Forget-free continual learning with winning subnetworks,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 734–10 750.
- [30] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03635*, 2018.
- [31] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias, “Brain-inspired replay for continual learning with artificial neural networks,” *Nature communications*, vol. 11, p. 4069, 2020.
- [32] A. Robins, “Catastrophic forgetting, rehearsal and pseudorehearsal,” *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.
- [33] 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.
- [34] 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.
- [35] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, “Dark experience for general continual learning: a strong, simple baseline,” *Advances in neural information processing systems*, vol. 33, pp. 15 920–15 930, 2020.
- [36] S. Channappayya, B. R. Tamma *et al.*, “Augmented memory replay-based continual learning approaches for network intrusion detection,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 17 156–17 169, 2023.
- [37] C. Chen, K. Ota, M. Dong, C. Yu, and H. Jin, “Human activity recognition-oriented incremental learning with knowledge distillation,” *Journal of Circuits, Systems and Computers*, vol. 30, no. 06, p. 2150096, 2021.
- [38] J. Von Oswald, C. Henning, B. F. Grewe, and J. Sacramento, “Continual learning with hypernetworks,” *arXiv preprint arXiv:1906.00695*, 2019.
- [39] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, “Learning to learn without forgetting by maximizing transfer and minimizing interference,” *arXiv preprint arXiv:1810.11910*, 2018.
- [40] J. Dey, A. Geisa, R. Mehta, T. M. Tomita, H. S. Helm, H. Xu, E. Eaton, J. Dick, C. E. Priebe, and J. T. Vogelstein, “Towards a theory of out-of-distribution learning,” *arXiv preprint arXiv:2109.14501*, 2021.
- [41] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” in *NIPS*, 2017.
- [42] D. Benavides-Prado, Y. S. Koh, and P. Riddle, “Measuring Cumulative Gain of Knowledgeable Lifelong Learners,” in *NeurIPS Continual Learning Workshop*, 2018, pp. 1–8.
- [43] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, “Don’t forget, there is more than forgetting: new metrics for continual learning,” *arXiv preprint arXiv:1810.13166*, 2018.
- [44] P. Judea, “What is gained from past learning,” *Journal of Causal Inference*, vol. 6, no. 1, 2018.
- [45] P. J. Bickel and K. A. Doksum, *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC, 2015.
- [46] S. Chakraborty, B. Uzcent, K. Ayush, K. Tanmay, E. Sheehan, and S. Ermon, “Efficient conditional pre-training for transfer learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4241–4250.
- [47] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.
- [48] B. Yuan and D. Zhao, “A survey on continual semantic segmentation: Theory,” *Challenge, Method and Application*, 2023.
- [49] W. Kuo and M. J. Zuo, *Optimal reliability modeling: principles and applications*. John Wiley & Sons, 2003.
- [50] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning (still) requires rethinking generalization,” *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.
- [51] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: John Wiley & Sons, Nov. 2012.
- [52] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.
- [53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. U. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008.
- [54] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018.

- [Online]. Available: <http://arxiv.org/abs/1810.04805>
- [55] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.
- [56] M. Denil, D. Matheson, and N. D. Freitas, “Narrowing the gap: Random forests in theory and in practice,” in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, 6 2014, pp. 665–673.
- [57] S. Athey, J. Tibshirani, and S. Wager, “Generalized random forests,” *Annals of Statistics*, vol. 47, no. 2, pp. 1148–1178, 2019.
- [58] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
- [59] Y. Freund, “Boosting a Weak Learning Algorithm by Majority,” *Inform. and Comput.*, vol. 121, no. 2, pp. 256–285, Sep. 1995.
- [60] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [61] Y. Amit and D. Geman, “Shape Quantization and Recognition with Randomized Trees,” *Neural Comput.*, vol. 9, no. 7, pp. 1545–1588, Oct. 1997.
- [62] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 144–161.
- [63] L. Wang, X. Zhang, Q. Li, J. Zhu, and Y. Zhong, “Coscl: Cooperation of small continual learners is stronger than a big one,” in *European Conference on Computer Vision*. Springer, 2022, pp. 254–271.
- [64] A. Prabhu, S. Sinha, P. Kumaraguru, P. Torr, O. Sener, and P. Dokania, “Randumb: Random representations outperform online continually learned representations,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 37 988–38 006, 2024.
- [65] A. Prabhu, P. H. Torr, and P. K. Dokania, “Gdumb: A simple approach that questions our progress in continual learning,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 524–540.
- [66] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, “Efficient lifelong learning with a-gem,” *arXiv preprint arXiv:1812.00420*, 2018.
- [67] P. Malviya, S. Chandar, and B. Ravindran, “Tag: Task-based accumulated gradients for lifelong learning,” *arXiv preprint arXiv:2105.05155*, 2021.
- [68] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” *CoRR*, vol. abs/1904.07734, 2019. [Online]. Available: <http://arxiv.org/abs/1904.07734>
- [69] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *University of Toronto*, 05 2012.
- [70] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas, and A. Thite, “Jakobovski/free-spoken-digit-dataset: v1. 0.8,” *Zenodo*, August, 2018.
- [71] W. Min, Z. Wang, Y. Liu, M. Luo, L. Kang, X. Wei, X. Wei, and S. Jiang, “Large scale visual food recognition,” *CoRR*, vol. abs/2103.16107, 2021.
- [72] F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras>, 2015.
- [73] A. Dosovitskiy, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [74] V. Lomonaco and D. Maltoni, “Core50: a new dataset and benchmark for continuous object recognition,” in *Conference on Robot Learning*. PMLR, 2017, pp. 17–26.
- [75] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Page-Caccia, I. H. Laradji, I. Rish, A. Lacoste, D. Vázquez *et al.*, “Online fast adaptation and knowledge accumulation (osaka): a new approach to continual learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 532–16 545, 2020.
- [76] C. Dwork, “Differential privacy: A survey of results,” in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.
- [77] I. van Rooij, M. Blokpoele, J. Kwisthout, and T. Wareham, *Cognition and Intractability: A Guide to Classical and Parameterized Complexity Analysis*. Cambridge University Press, Apr. 2019.
- [78] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.
- [79] Y. Bulatov, “<http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html>,” 2011.
- [80] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [81] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, “Explaining the success of adaboost and random forests as interpolating classifiers,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1558–1590, 2017.
- [82] R. Caruana and A. Niculescu-Mizil, “An Empirical Comparison of Supervised Learning Algorithms,” in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML ’06. New York, NY, USA: ACM, 2006, pp. 161–168.
- [83] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, “Ensemble selection from libraries of models,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 18.

VIII. BIOGRAPHY SECTION



Joshua T. Vogelstein is an Associate Professor in the Department of Biomedical Engineering at Johns Hopkins University, with joint appointments in Applied Mathematics and Statistics, Computer Science, Electrical and Computer Engineering, Neuroscience, and Biostatistics. His research focuses primarily on the intersection of natural and artificial intelligence. His lab develops and applies high-dimensional nonlinear machine learning methods to biomedical big data science challenges. He has published about 200 papers in prominent scientific and engineering

venues, with $> 12,000$ citations and an h-index > 45 . His group is one of the few in the world that regularly publishes in both top scientific (e.g., Nature, Science, Cell, PNAS, eLife) and top artificial intelligence (e.g., JMLR, Neurips, ICML) venues. His group has received funding from the Transformative Research Award from NIH, the NSF CAREER award, Microsoft Research, and many other government, for-profit and nonprofit organizations. He has advised over 60 trainees, and taught about 200 students in his eight years as faculty. In addition to his academic work, he co-founded Global Domain Partners, a quantitative hedge fund that was acquired by Mosaic Investment Partners in 2012, and software startup Gigantum, which was acquired by nVidia in early 2022. He lives in the Chesapeake Bay Watershed with his beloved eternal wife and our three children.



Jayanta Dey received his B.Sc. and M. Sc. degrees in electrical and electronic engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2017 and 2019, respectively with a focus on ultrasound imaging system and signal processing. Subsequently, he earned his M.Sc. and PhD in biomedical engineering from Johns Hopkins University under the supervision of Dr. Joshua T. Vogelstein in 2024 with a focus on out-of-distribution learning. This work is a major part of his PhD thesis. Currently, he is working as

a postdoctoral fellow under the supervision of Dr. Dhireesha Kudithipudi in NuAI lab at UTSA with a focus on the temporal aspects of out-of-distribution learning. His research interest includes making artificial intelligence less artificial and more natural.



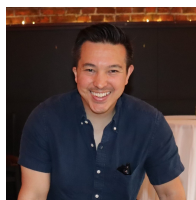
Hayden S. Helm is a researcher specializing in statistical pattern recognition and machine learning. He is the founder and principal researcher at Helivan Research, based in San Francisco, California. He also holds a position as a researcher at Nomic AI.



Will LeVine is a researcher at Microsoft's Mixed Reality division. He did his Master's at Johns Hopkins but dropped out of college. His current interests lie at the intersection of interpretability, reliability, and multi-modal learning.



Ronak D. Mehta is a Ph.D. candidate at the Department of Statistics, University of Washington, advised by Zaid Harchoui. He completed his undergraduate and Master's at Johns Hopkins University under the supervision of Joshua T. Vogelstein and Carey Priebe. His research interests lie broadly in mathematical optimization, distributional shift, and distributional robustness.



Tyler M. Tomita is a data scientist and machine learning researcher at the Johns Hopkins University Applied Physics Laboratory. He received his B.S. in Biomedical Engineering from the University of California, Davis and his Ph.D in Biomedical Engineering at Johns Hopkins University under the supervision of Dr. Joshua Vogelstein. His interests are in flexible, robust, and data-efficient machine learning solutions.



Haoyin Xu is a PhD student at The Johns Hopkins University, Department of Biomedical Engineering. His research fields include biomedical data science and machine learning. He is currently working on random forests based hypothesis testing and online learning methods.



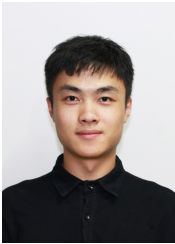
Ali Geisa is a software engineer at Vendelux. He completed his undergraduate and Master's at Johns Hopkins University, and was part of the Neurodata lab with Dr. Joshua T. Vogelstein. His research interests lie broadly in machine learning and algorithm development.



Qingyang (Alice) Wang is a Ph.D. candidate in the Department of Neuroscience at Johns Hopkins University, where she is advised by Dr. Joshua Vogelstein and Dr. Carey Priebe. She earned her Bachelor of Science degree from the Hong Kong University of Science and Technology, with a double major in Biochemistry and Cell Biology, and Computer Science. Her research focuses on the intersection of neuroscience and AI, exploring how insights from the brain can advance AI and how AI models can deepen our understanding of the brains.



Gido M. van de Ven is currently a postdoctoral researcher at the KU Leuven in Belgium, where he studies continual learning from both a deep learning and a cognitive science perspective. At the time of writing this paper, he was a postdoctoral researcher at the Baylor College of Medicine in Houston and a visiting researcher at the University of Cambridge.



Chenyu Gao is a PhD student in Electrical and Computer Engineering at Vanderbilt University. His research focuses on image processing and computer vision for medical image analysis. He obtained a master's degree in Biomedical Engineering from Johns Hopkins University.



Carey E. Priebe (Senior Member, IEEE) received the B.S. degree in mathematics from Purdue University, West Lafayette, IN, USA, in 1984, the M.S. degree in computer science from San Diego State University, San Diego, CA, USA, in 1988, and the Ph.D. degree in information technology (computational statistics) from George Mason University, Fairfax, VA, USA, in 1993. From 1985 to 1994, he was a mathematician and scientist with the US Navy research and development laboratory system. Since 1994, he has been a Professor with the Department of Applied Mathematics and Statistics, Johns Hopkins University. His research interests include computational statistics, kernel and mixture estimates, statistical pattern recognition, model selection, and statistical inference for high-dimensional and graph data. He is an Elected Member of the International Statistical Institute, a Fellow of the Institute of Mathematical Statistics, and a Fellow of the American Statistical Association.



Weiwei Yang works in resource efficient alt-SGD ML methods inspired by biological learning. The applied research group, she leads, aims to democratize AI by addressing issues of sustainability, robustness, scalability, and efficiency in ML. Her group has applied ML to address social issues such as countering human trafficking and to energy grid stabilizations.



Bryan Tower has been working with machine learning, information retrieval, and graph learning for over twenty years. Most recently he has been working for Microsoft Research on Large Foundational Models.



Jonathan Larson is a Senior Principal Data Architect at Microsoft Research working in Special Projects. He currently leads a research team focused on the intersection of graph machine learning, LLM memory representations, and LLM orchestration.



Christopher M. White is partner and managing director of special projects at Microsoft. He leads research teams with world-class specialists solving highly uncertain, complex problems. His group builds technologies to benefit society, including tools for digital safety, plurality, and evidence-based policy.

APPENDIX

A. Decomposition of Transfer

Transfer can be decomposed into Forward Transfer and Backward Transfer:

$$\text{Transfer}^t(f) = \log \frac{\mathcal{E}_f^t(\mathbf{S}^t)}{\mathcal{E}_f^t(\bigcup_{t'=1}^T \mathbf{S}^{t'})} \quad (7)$$

$$= \log \frac{\mathcal{E}_f^t(\mathbf{S}^t)}{\mathcal{E}_f^t(\bigcup_{t'=1}^t \mathbf{S}^{t'})} + \log \frac{\mathcal{E}_f^t(\bigcup_{t'=1}^t \mathbf{S}^{t'})}{\mathcal{E}_f^t(\bigcup_{t'=1}^T \mathbf{S}^{t'})} \quad (8)$$

$$= \text{Forward Transfer}^t(f) + \text{Backward Transfer}^t(f). \quad (9)$$

We say that an algorithm f has transferred to task t from all the tasks up to T if and only if

$$\text{Transfer}^t(f) > 0. \quad (10)$$

B. Representation Ensembling Algorithms

1) *Model Architecture*: In this paper, we proposed two representation ensembling algorithms, Simple Lifelong Learning Networks (SILLY-N) and Simple Lifelong Learning Forests (SILLY-F). The two algorithms differ in their details of how to update encoders and channels, but abstracting a level up they are both special cases of the same procedure. Let SILLY-X refer to any possible representation algorithm. Algorithms 1, 2, 3, and 4 provide pseudocode for adding encoders, updating channels, and doing inference for any SILLY-X algorithm.

2) *Data Preparation*: Whenever the learner gets access to a new task data, we use Algorithm 1 to train a new encoder for the corresponding task. We split the data into two portions — in-task set and held out or out-of-bag set.

3) *Training Procedures*: In-task set is used to learn the encoder and the indices of the out-of-bag (OOB) data which is returned by Algorithm 1 to be used by Algorithm 2 to learn the channel for the corresponding task. Note that we push the OOB data through the in-task encoder and the whole dataset through the cross-task encoders to update the channel, i.e., learn the posteriors according to the new encoder (see Algorithm 3). Finally, Algorithm 4 does inference on a new sample. Given the task identity, we use the corresponding channel to get the average estimated posterior and predict the class label as the arg max of the estimated posteriors.

Algorithm 1 Add a new SILLY-X encoder for a task. OOB = out-of-bag.

Require:

- (1) t \triangleright current task number
- (2) $\mathcal{D}_n^t = (\mathbf{x}^t, \mathbf{y}^t) \in \mathbb{R}^{n \times p} \times \{1, \dots, K\}^n$ \triangleright training data for task t

Ensure:

- (1) u_t \triangleright an encoder trained on task t
- (2) \mathcal{I}_{OOB}^t \triangleright a set of the indices of OOB data
- 1: **function** SILLY-X.FIT($t, (\mathbf{x}^t, \mathbf{y}^t)$)
- 2: $u_t, \mathcal{I}_{OOB}^t \leftarrow \text{encoder.fit}(\mathbf{x}^t, \mathbf{y}^t)$ \triangleright train an encoder on training data partitioned into in-bag and OOB samples
- 3: **return** u_t, \mathcal{I}_{OOB}^t
- 4: **end function**

Algorithm 2 Add a new SILLY-X channel for the current task.

Require:

- (1) t \triangleright current task number
- (2) $\mathcal{U} = \{u_{t'}\}_{t'=1}^t$ \triangleright the set of encoders
- (3) $\mathcal{D}_n^t = (\mathbf{x}_t, \mathbf{y}_t) \in \mathbb{R}^{n \times p} \times \{1, \dots, K\}^n$ \triangleright training data for task t
- (4) \mathcal{I}_{OOB}^t \triangleright a set of the indices of OOB data for the current task

Ensure: v_t \triangleright channel for task t

- 1: **function** SILLY-X.ADD_CHANNEL($t, u_t, (\mathbf{x}_t, \mathbf{y}_t), \mathcal{I}_{OOB}^t$)
- 2: $v_t \leftarrow u_t.\text{add_channel}((\mathbf{x}_t, \mathbf{y}_t), \mathcal{I}_{OOB}^t)$ \triangleright add the new in-task channel using OOB data
- 3: **for** $t' = 1, \dots, t-1$ **do**
- 4: $v_t \leftarrow u_{t'}.\text{update_channel}(\mathbf{x}_t, \mathbf{y}_t, v_t)$ \triangleright update the channel for task t using the old encoders
- 5: **end for**
- 6: **return** v_t
- 7: **end function**

Algorithm 3 Update SILLY-X channel for the previous tasks.

Require:

- (1) t \triangleright current task number
- (2) u_t \triangleright encoder for the current task
- (3) $\mathcal{D} = \{\mathcal{D}^{t'}\}_{t'=1}^{t-1}$ \triangleright training data for old tasks
- (4) $\mathcal{V} = \{v_{t'}\}_{t'=1}^{t-1}$ \triangleright set of all previous task voters

Ensure: $\mathcal{V} = \{v_{t'}\}_{t'=1}^{t-1}$

- 1: **function** SILLY-X.UPDATE_OLD_CHANNEL($t, u_t, \mathcal{D}, \mathcal{V}$)
- 2: **for** $t' = 1, \dots, t-1$ **do**
- 3: $v_{t'} \leftarrow u_t.\text{update_channel}(\mathcal{D}^{t'}, v_{t'})$ \triangleright update the old task channels
- 4: **end for**
- 5: **return** \mathcal{V}
- 6: **end function**

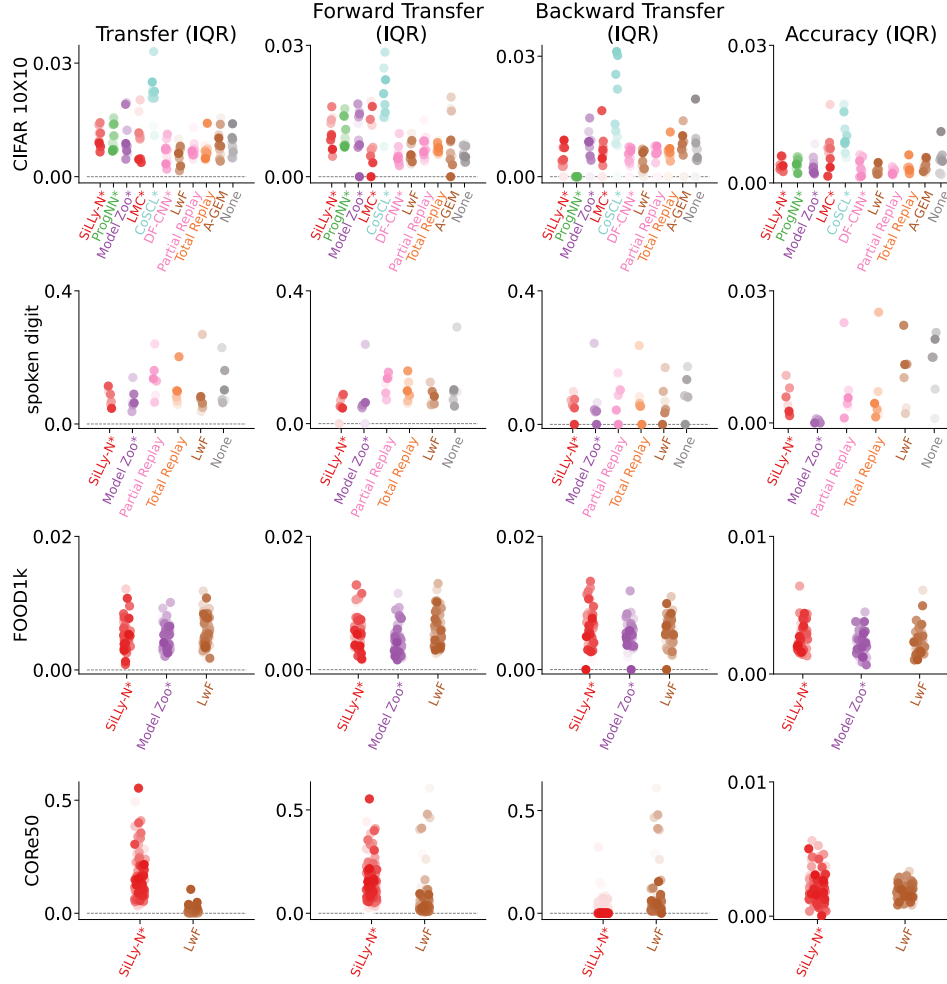


Fig. 1. Error bars (interquartile range, i.e., IQR) for each dot in Figure 1 on different vision and speech benchmark datasets. Error bars for different performance statistics is negligible in comparison with the median performance shown in the main text Figure 1.

Algorithm 4 Predicting a class label using SYNX.

Require:

- (1) $\mathbf{x} \in \mathbb{R}^p$ \triangleright test datum
- (2) t \triangleright task identity associated with \mathbf{x}
- (3) \mathcal{U} \triangleright set of all t encoders
- (4) v_t \triangleright channel for task t

Ensure: \hat{y} \triangleright a predicted class label

```

1: function  $\hat{y} = \text{SiLly-X.PREDICT}(t, \mathbf{x}, v_t)$ 
2:   for  $t' = 1, \dots, t$  do  $\triangleright$  get the output  $\tilde{\mathbf{x}} = \{\tilde{\mathbf{x}}_{t'}\}_{t'=1}^t$ 
     from all the encoders
3:      $\tilde{\mathbf{x}}_{t'} \leftarrow v_{t'}.encode(\mathbf{x})$ 
4:   end for
5:    $\hat{\mathbf{p}} \leftarrow v_t.predict\_proba(\tilde{\mathbf{x}})$   $\triangleright$   $\hat{\mathbf{p}}$  is a  $K_t$ -dimensional
     posterior vector
6:    $\hat{y} = \arg \max(\hat{\mathbf{p}})$   $\triangleright$  find the index of the elements in
     the vector  $\hat{\mathbf{p}}$  with maximum value
7:   return  $\hat{y}$ 
8: end function

```

TABLE I

HYPERPARAMETERS FOR SiLly-N IN CIFAR 10X10, FIVE DATASETS, SPLIT MINI-IMAGENET, FOOD1k EXPERIMENTS. NOTE THAT WE USE THE SAME HYPERPARAMETERS FOR ALL THE EXPERIMENTS.

Hyperparameters	Value
optimizer	Adam
learning rate	3×10^{-4}
max_samples (OOB split)	0.67
n_estimators (decision forest channel)	20

TABLE II

HYPERPARAMETERS FOR SiLly-F IN TABULAR CIFAR 10X10 EXPERIMENTS.

Hyperparameters	Value
n_estimators	10
max_depth	30
max_samples (OOB split)	0.67
min_samples_leaf	1

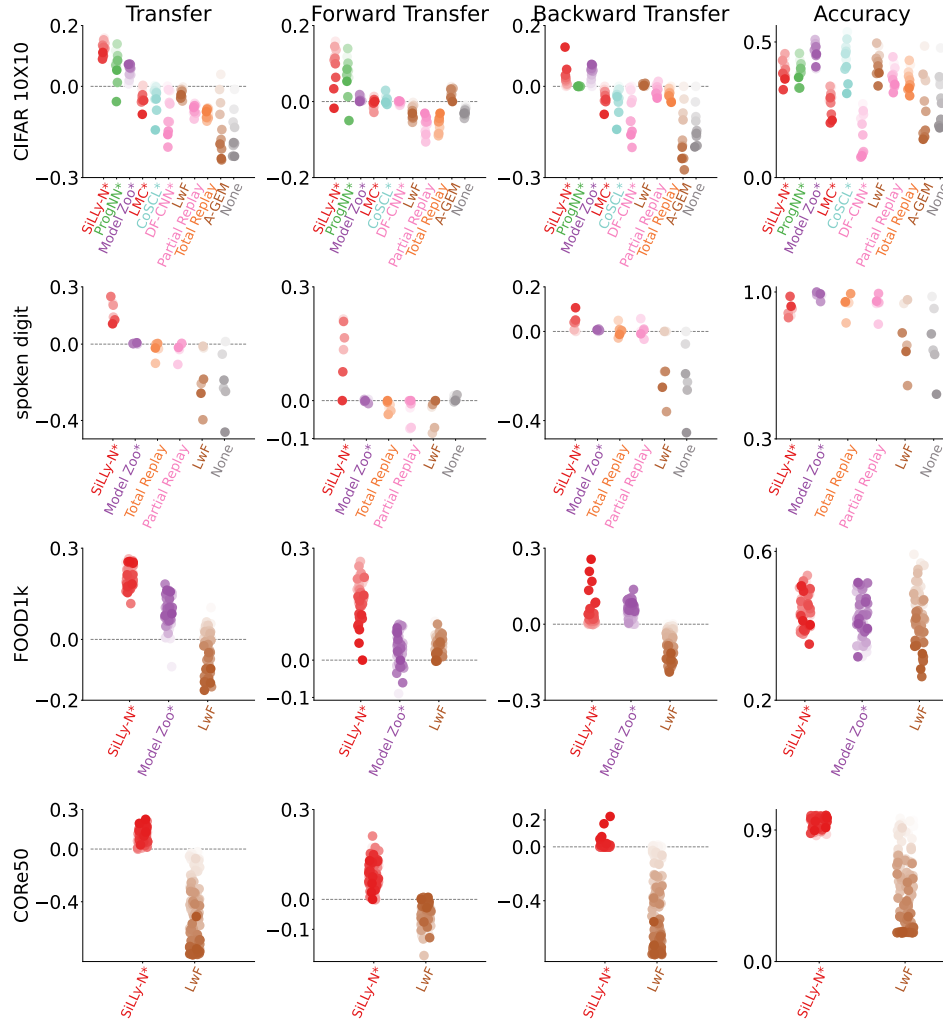


Fig. 2. **Performance summary on vision and audition benchmark datasets using Veniat’s [25]’s statistics.** See Figure 1 for caption details. Note that the results here look nearly identical other than the y-axis labels.

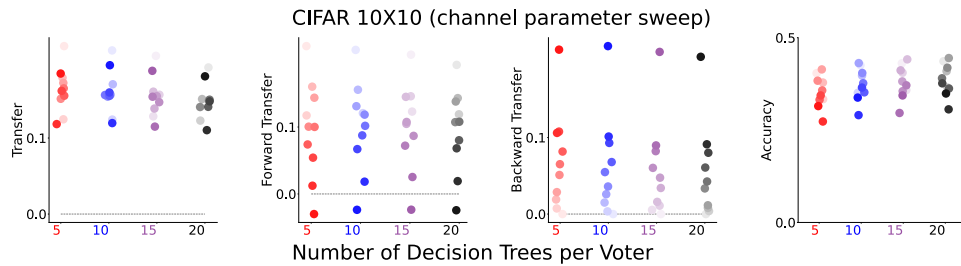


Fig. 3. **Performance of SiLLy-N on CIFAR 10x10 remains nearly unchanged for different number of decision trees per channel.**

C. Reference Algorithm Implementation Details

The same network architecture was used for all baseline deep learning methods. Following the work in [31], the ‘base network architecture’ consisted of five convolutional layers followed by two fully connected layers each containing 2000 nodes with ReLU non-linearities and a softmax output layer. The convolutional layers had 16, 32, 64, 128 and 254 channels, they used batch-norm and a ReLU non-linearity, they had a 3x3 kernel, a padding of 1 and a stride of 2 (except the first layer, which had a stride of 1). This architecture was used

with a multi-headed output layer (i.e., a different output layer for each task) for all algorithms using a fixed-size network. For ProgNN and DF-CNN the same architecture was used for each column introduced for each new task, and in our SiLLy-N this architecture was used for the transformers u_t (see above). In these implementations, ProgNN and DF-CNN have the same architecture for each column introduced for each task. Among the reference algorithms, EWC, O-EWC, LwF, SI, TOTAL REPLAY and PARTIAL REPLAY results were produced using the repository <https://github.com/GMvandeVen/>

progressive-learning-pytorch. For `PROGNN` and `DF-CNN` we used the code provided in <https://github.com/Lifelong-ML/DF-CNN>. For all other reference algorithms, we modified the code provided by the authors to match the deep net architecture as mentioned above and used the default hyperparameters provided in the code.

D. Training Time Complexity Analysis

We use the soft-O notation \tilde{O} to quantify complexity [77]. Letting n be the sample size and T be the number of tasks, we write that the capacity, space or time complexity of a lifelong learning algorithm is $f(n, t) = \tilde{O}(g(n, T))$ when $|f|$ is bounded above asymptotically by a function g of n and T up to a constant factor and polylogarithmic terms. For simplifying the calculation, we make the following assumptions:

- 1) Each task has the same number of training samples.
- 2) Capacity grows linearly with the number of trainable parameters in the model.
- 3) The number of epochs is fixed for each task.
- 4) For the algorithms with dynamically expanding capacity, we assume the worst case scenario where an equal amount of capacity is added to the hypothesis with an additional task.

Assumption 3 enables us to write time complexity as a function of the sample size. Table I summarizes the capacity, space and time complexity of several reference algorithms, as well as our `SILLY-N` and `SILLY-F`.

Lifelong learning methods are parametric if they have a representational capacity which is invariant to sample size and task number. Although the space complexity of some of these algorithms grow (because the number of the constraints stored by the algorithms grows, or they continue to store more data), their capacity is fixed. Thus, given a sufficiently large number of tasks with increasing complexity, in general, eventually all parametric methods will catastrophically forget. `EWC` [11], `ONLINE EWC` [12], `SI` [14], and `LWF` [15] are all examples of parametric lifelong learning algorithms. Our fixed resource algorithms are also parametric.

Lifelong learning methods are semi-parametric if they have a representational capacity which grows slower than sample size. For example, if T is increasing slower than n (e.g., $T \propto \log n$), then algorithms whose capacity is proportional to T are semi-parametric. `PROGNN` [22] is semi-parametric, nonetheless, its space complexity is $\tilde{O}(T^2)$ due to the lateral connections. Moreover, the time complexity for `PROGNN` also scales quadratically with n when $n \propto T$. Thus, an algorithm that literally stores all the data it has ever seen, and retrains a fixed size network on all those data with the arrival of each new task, would have smaller space complexity and the same time complexity as `PROGNN`. `DF-CNN` [23] improves upon `PROGNN` by introducing a “knowledge base” with lateral connections to each new column, thereby avoiding all pairwise connections. Because these semi-parametric methods have a fixed representational capacity per task, they will either lack the representation capacity to perform well given sufficiently complex tasks, and/or will waste resources for very simple tasks.

Lifelong learning methods are non-parametric if they have a representational capacity which *grow in proportion* to the number of tasks or data samples. Table I shows the Indian Buffet Process for Weight Factors (`IBP-WF`) is a notable non-parametric approach alongside `SILLY-F`.

Our proposed approaches, `SILLY-N` and `SILLY-F`, as we will discuss in details in Section IV, eliminate lateral connections between the columns of the network, thus reducing the complexity of the space to $\tilde{O}(T)$. Moreover, our proposed approaches can adapt flexibly to any of the three categories based on the constraints of the application environment, as illustrated in Table I.

1) *Complexity analysis*: Consider a lifelong learning environment with T tasks each with n' samples, i.e., total training samples, $n = n'T$. For all the algorithm with time complexity $\tilde{O}(n)$, the training time grows linearly with more training samples. We discuss all other algorithms with non-linear time complexity below.

a) *EWC*: Consider the time required to train the weights for each task in `EWC` is $k_c n'$ and each task adds additional $k_l n'$ time from the regularization term. Here, k_c and k_l are both constants. Therefore, time required to learn all the T tasks can be written as:

$$\begin{aligned}
 & k_c n' + (k_c n' + k_l n') + \dots + (k_c n' + (T-1)k_l n') \\
 &= k_c n' T + k_l n' \sum_{t=1}^{T-1} t \\
 &= k_c n' T + k_l n' \frac{T(T-1)}{2} \\
 &= k_c n' + 0.5 k_l n' T - 0.5 k_l n' \\
 &= \tilde{O}(nT).
 \end{aligned} \tag{11}$$

b) *Total Replay*: Consider the time to train the model on n' samples is $k_c n'$. Therefore, time required to learn all the T tasks can be written as:

$$\begin{aligned}
 & k_c n' + k_c (n' + n') + \dots + k_c n' T \\
 &= k_c n' \sum_{t=1}^T t \\
 &= k_c n' \frac{T(T+1)}{2} \\
 &= 0.5 k_c n' T + 0.5 k_c n' \\
 &= \tilde{O}(nT)
 \end{aligned} \tag{12}$$

c) *PROGNN*: Consider the time required to train each column in `PROGNN` is $k_c n'$ and each lateral connection can be learned with time $k_l n'$. Therefore, time required to learn all the T tasks can be written as:

TABLE III
BENCHMARK DATASET DETAILS.

Experiment	Dataset	Training samples	Testing samples	Dimension
CIFAR 10X10	CIFAR 100	5000	10000	$3 \times 32 \times 32$
5-dataset	CIFAR-10	50000	10000	$3 \times 32 \times 32$ (resized)
	MNIST	60000	10000	
	SVHN	73257	26032	
	notMNSIT	16853	1873	
	Fashion-MNIST	60000	10000	
Split Mini-Imagenet	Mini-Imagenet	48000	12000	$3 \times 84 \times 84$
FOOD1k 50X20	Food1k	60000	99682	$3 \times 50 \times 50$ (resized)
Spoken Digit	Spoken Digit	1650	1350	28×28 (processed and resized)

$$\begin{aligned}
& k_c n' + (k_c n' + k_l n') + \dots + (k_c n' + (T-1)k_l n') \\
&= k_c n' T + k_l n' \sum_{t=1}^{T-1} t \\
&= k_c n' T + k_l n' \frac{T(T-1)}{2} \\
&= k_c n + 0.5 k_l n T - 0.5 k_l n \\
&= \tilde{\mathcal{O}}(nT)
\end{aligned} \tag{13}$$

E. Simulation Experiment Details

In each simulation, we constructed an environment with two tasks. For each, we sample 750 times from the first task, followed by 750 times from the second task. These 1,500 samples comprise the training data. We sample another 1,000 hold out samples to evaluate the algorithms. For **SILLY-N**, we have used a deep network (DN) architecture with two hidden layers each having 10 nodes. Similarly, for **SILLY-N** experiments we did 100 repetitions and reported the results after smoothing it using moving average with a window size of 5.

Gaussian XOR is two class classification problem with equal class priors. Conditioned on being in class 0, a sample is drawn from a mixture of two Gaussians with means $\pm [0.5, 0.5]^\top$, and variances proportional to the identity matrix. Conditioned on being in class 1, a sample is drawn from a mixture of two Gaussians with means $\pm [0.5, -0.5]^\top$, and variances proportional to the identity matrix. Gaussian XNOR is the same distribution as Gaussian XOR with the class labels flipped. Rotated XOR (R-XOR) rotates XOR by θ° degrees.

This simulation setup facilitates the manipulation of task overlap, allowing for an examination of the transfer properties of our proposed approach under different levels of task similarity (see main text).

F. Real Data Extended Experiments and Details

This section contains extended results on algorithms not shown in the main text (see Appendix Figure 5). FOOD1k and Mini-Imagenet datasets were obtained from <https://www.kaggle.com/datasets/whitemoon/minimagenet> and <https://github.com/pranshu28/TAG>, respectively.

a) *Split Mini-Imagenet*: In this experiment, we have used the **Mini-Imagenet** dataset [67]. The dataset was split into 20 tasks with 5 classes each. Each task has 2400 training samples and 600 testing samples. As shown in Figure 5 right column, we get positive forward and backward transfer for **SILLY-N**. However, although samples per task is lower compared to that of 5-dataset, it is still quite high. Hence, **MODEL ZOO** outperforms all the algorithms in this experiment.

b) *5-dataset*: In this experiment, we have used **5-dataset** [67]. It consists of 5 tasks from five different datasets: CIFAR-10 [69], MNIST, SVHN [78], notMNIST [79], Fashion-MNIST [80]. All the monochromatic images were converted to RGB format, and then resized to $3 \times 32 \times 32$. As shown in Appendix Table III, training samples per task in 5-dataset is relatively higher than that of low data regime typically considered in lifelong learning setting. However, as shown in Figure 5 left column, **SILLY-N** show less forgetting than most of the reference algorithms. On the other hand, **MODEL ZOO** shows comparatively better performance in relatively high task data size setup. Recall that **SILLY-N** is based on bagging, and **MODEL ZOO** is based on boosting. It is well known that boosting often outperforms bagging when sample sizes are large ².

c) *Overlapping Task Experiment*: We considered the setting where each task is defined by a random sampling of 10 out of 100 classes with replacement in CIFAR 10x10. This environment is designed to demonstrate the effect of having overlapping tasks, which is a common property of real world lifelong learning tasks. Appendix Figure 4 shows positive transfer from other tasks to Task 1 for **SILLY-F** and **SILLY-N**.

d) *Spoken Digit Experiment Details*: In this experiment, we used the **Spoken Digit** dataset provided in <https://github.com/Jakobovski/free-spoken-digit-dataset>. The dataset contains audio recordings from six different speakers with 50 recordings for each digit per speaker (3000 recordings in total). The experiment was set up with six tasks where

²Authors in [81] shows that both bagging and boosting asymptotically converge to the Bayes optimal solution. However, for finite sample size and similar model complexity, we empirically find bagging approach to lifelong learning performs better than that of boosting when the training sample size is low whereas boosting performs better on large training sample size (See Figure 1). This is consistent with similar results in single task learning [43, 82, 83]

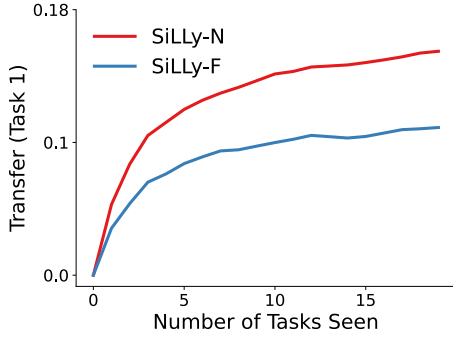


Fig. 4. SiLLy-N and SiLLy-F transfer knowledge effectively when tasks share common classes. Each task is a random selection of 10 out of the 100 CIFAR-100 classes. Both SiLLy-F and SiLLy-N demonstrate monotonically increasing transfer efficiency for up to 20 tasks.

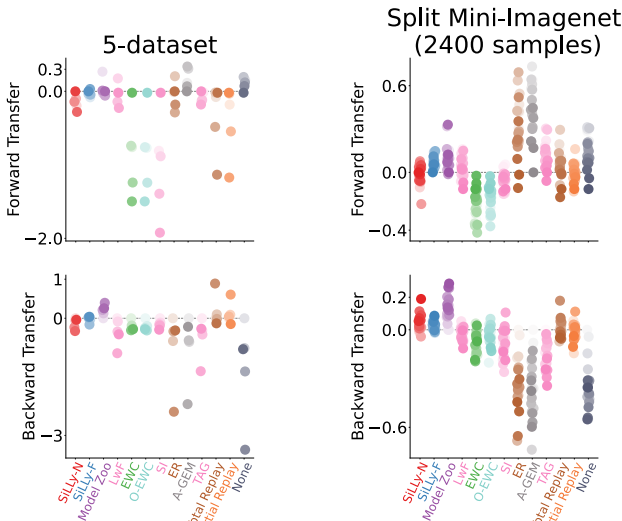


Fig. 5. Performance of different lifelong learners on two vision datasets..

each task contains recordings from only one speaker. For each recording, a spectrogram was extracted using Hanning windows of duration 16 ms with an overlap of 4 ms between the adjacent windows. The spectrograms were resized down to 28×28 . The extracted spectrograms from eight random recordings of ‘5’ for six speakers are shown in Figure 6. For each Monte Carlo repetition of the experiment, spectrograms extracted for each task were randomly divided into 55% train and 45% test set. The experiment is summarized in Figure 7. Note that we could not run the experiment on other 5 reference algorithms using the code provided by their authors.

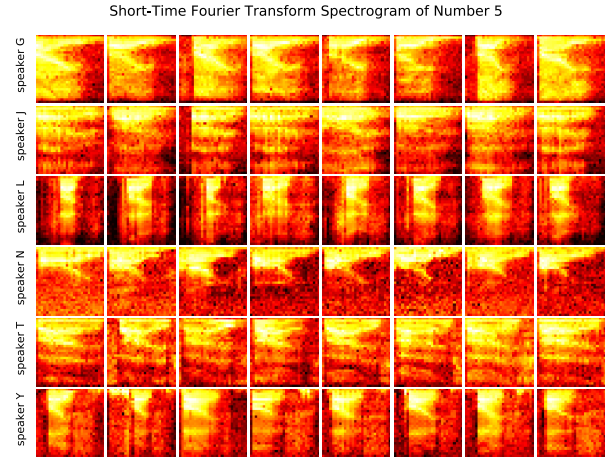


Fig. 6. Spectrogram extracted from eight different recordings of six speakers uttering the digit ‘five’.

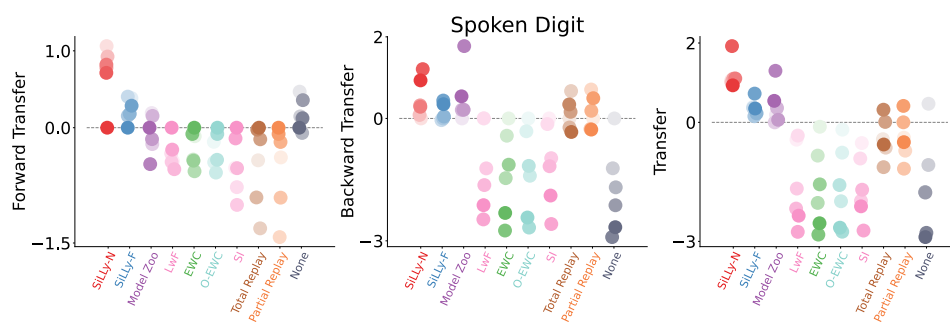


Fig. 7. **Extended results on the Spoken Digit experiments.** This plot contains algorithms not shown in main text Figure 1.