A General Approach to Progressive Learning

Joshua T. Vogelstein[1,†,*], Hayden S. Helm[1,†], Ronak D. Mehta[1], Jayanta Dey[1], Weiwei Yang[2],
Bryan Tower[2], Will LeVine[3], Jonathan Larson[2], Chris White[2], and Carey E. Priebe[1]

Abstract. In biological learning, data is used to improve performance on the task at hand, while simultaneously improving performance on both previously encountered tasks and as yet unconsidered future tasks. In contrast, classical machine learning starts from a blank slate, or tabula rasa, using data only for the single task at hand. While typical transfer learning algorithms can improve performance on future tasks, their performance degrades upon learning new tasks. Many recent approaches have attempted to mitigate this issue, called catastrophic forgetting, to *maintain* performance given new tasks. But striving to avoid forgetting sets the goal unnecessarily low: the goal of progressive learning, whether biological or artificial, is to *improve* performance on all tasks (including past and future) with any new data. We propose a general approach to progressive learning that ensembles representations, rather than learners. We show that ensembling representations—including representations learned by decision forests or neural networks—enables both forward and backward transfer on a variety of simulated and real data tasks, including vision, language, and adversarial tasks. This work suggests that further improvements in progressive learning may follow from a deeper understanding of how biological learning achieves such high degrees of efficiency.

Learning from data is the process by which an intelligent system improves performance on a given task by leveraging data [24]. In biological learning, learning is progressive, continually building on past knowledge and experiences, improving on many tasks given data associated with any task. For example, learning a second language often improves performance in an individual's native language [40].

In classical machine learning, the system often starts with essentially zero knowledge, a "tabula rasa", and is optimized for a single task [37, 36]. While it is relatively easy to *simultaneously* optimize for multiple tasks (multi-task learning) [8], it has proven much more difficult to *sequentially* optimize for multiple tasks [34, 35]. 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 [22, 21]. This is the opposite of many biological learning settings, such as the second language learning setting mentioned above.

In the past 30 years, a number of sequential learning methods have attempted to overcome catastrophic forgetting. These approaches naturally fall into one of two camps. In the first, the system adds (or builds) resources as new data arrive [30, 19]. Biologically, this corresponds to development, where brains grow by adding neurons and/or synapses. In the second, the system has fixed resources, and so must reallocate resources (essentially compressing representations) in order to incorporate new knowledge [16, 20, 39, 20, 32]. Biologically, this corresponds to adulthood, where brains have a nearly fixed or decreasing number of neurons and synapses.

Approaches from both camps demonstrate continual [27], or lifelong learning. In particular, they can sometimes learn new tasks while not catastrophically forgetting old tasks. However, as we will show, none of the previous algorithms are able to meaningfully transfer knowledge across tasks—either forward or backward—both key capabilities in progressive learning.

We present a general approach to progressive learning that we call "ensembling representations". Ensembling representation algorithms sequentially learn a representation for each task, and ensemble both old and new representations for all future decisions. The task-specific representations can be learned using any desirable mechanism. We implement two complementary ensembling representation algorithms, one based on ensembling decision forests (Lifelong Forests), and another based on ensembling deep neural networks (Lifelong Networks). Simulations illustrate the limitations and capabilities of these approaches, including performance properties in the presence of adversarial tasks. We then

---

[1]Johns Hopkins University (JHU), [2]Microsoft Research, [3]JHU Applied Physics Lab
[†] denotes equal contribution, [*] corresponding author: jovo@jhu.edu

demonstrate the capabilities of ensembling representation approaches on multiple real datasets, including both vision and language applications. Although ensembling representations are in the resource building camp mentioned above, we illustrate that they can continue to develop, converting from the juvenile resource building state to the adult resource recruiting state, while maintaining their progressive learning capabilities.

1 Background

1.1 Classical Machine Learning Classical supervised learning [25] considers random variables $(X, Y) \sim P_{X,Y}$, where X is an $\mathcal{X}$-valued input, Y is a $\mathcal{Y}$-valued response, and $P_{X,Y} \in \mathcal{P}_{X,Y}$ is the joint distribution of (X, Y). Given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, the goal is to find the hypothesis, or predictor, $h : \mathcal{X} \to \mathcal{Y}$ that minimizes expected loss, or *risk*. $R(h) = \mathbb{E}_{X,Y} [\ell(h(X), Y)]$. The minimum risk depends on the unknown $P_{X,Y}$.

A learning algorithm (or rule) is a sequence of functions, (f_n), where each f_n maps from n training samples, $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$, to a hypothesis in a class of hypotheses $\mathcal{H}$, $f_n : (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{H}$. A learning algorithm is evaluated on its generalization error $\mathcal{E}$ (or expected risk) at a particular sample size n: $\mathcal{E}(f_n) = \mathbb{E}[R(f_n(\mathcal{D}_n))]$, where the expectation is taken with respect to $\mathcal{D}_n$. The goal is to find a f_n that has a small generalization error assuming each (X, Y) pair is independent and identically distributed from some true but unknown $P_{X,Y}$ [25].

1.2 Lifelong Learning Lifelong learning generalizes classical machine learning in two ways: (i) instead of one task, there is an environment of (possibly infinitely) many tasks, and (ii) data arrive sequentially, rather than in batch mode. This setting requires that an algorithm can generalize to "out-of-past-task" examples. The goal in lifelong learning, given data from a new task, is to use all the data from previous tasks to achieve lower generalization error on this new task, and use the new data to improve the generalization error on all the previous tasks. Note that this is much stronger than simply avoiding catastrophic forgetting, which would mean that generalization on past tasks does not degrade. We define a progressive learning system as one that *improves* performance on past tasks given new data.

Previous work in lifelong learning falls loosely into two algorithmic frameworks for multiple tasks: (i) learning models with parameters specific to certain tasks and parameters shared across tasks [31], and (ii) decreasing the "size" of the hypothesis class with respect to the amount of training data [13]. Some approaches additionally store or replace (rehearse) previously encountered data to reduce forgetting [16]. The literature, however, has not yet codified general evaluation criteria to demonstrate lifelong learning on sequences of tasks [28].

2 Evaluation Criteria We define *transfer efficiency* as the ratio of the generalization error of (i) an algorithm that has learned only from data associated with a given task, to (ii) (ii) the same learning algorithm that also has access to other data. Transfer efficiency is akin to relative efficiency from classical statistics [3]. Whereas relative efficiency typically compares two different estimators for a given sample size, transfer efficiency compares the same estimator on two different datasets (one a subset of the other). Let R^t be the risk associated with task t. Let $\mathcal{E}^t(f_n^t)$ denote the error on task t of the algorithm that consumes only task t data: $\mathcal{E}^t(f_n^t) = \mathbb{E}[R^t(f_n^t(\mathcal{D}_n))]$. Similarly, let $\mathcal{E}^t(f_n)$ denote the error on task t of the algorithm that consumes all the data, $\mathcal{E}^t(f_n) = \mathbb{E}[R^t(f_n(\mathcal{D}_n))]$.

Definition 1 (Transfer Efficiency). *The transfer efficiency of algorithm f for given task t with sample size n is* $\mathsf{TE}^t(f_n) := \frac{\mathcal{E}^t(f_n^t)}{\mathcal{E}^t(f_n)}$. *Algorithm f_n transfer learns if and only if* $\mathsf{TE}^t(f_n) > 1$.

To evaluate a progressive learning algorithm while respecting the streaming nature of the tasks, it is convenient to consider two extensions of transfer efficiency. Transfer efficiency admits a factorization into forward and backward transfer efficiency: $\mathsf{TE}^t(f_n) = \mathsf{FTE}^t(f_n) \times \mathsf{BTE}^t(f_n)$.

Forward transfer efficiency is the ratio of (i) the generalization error of the algorithm with access only to task t data, to (ii) generalization error of the algorithm with sequential access to the data up to and including the last observation from task t. Thus, this quantity measures the relative effect of previously

seen out-of-task data on the performance on task t. Formally, let $N^t = \max\{i : T_i = t\}$, be the index of the last occurrence of task t in the data sequence. Let $\mathcal{D}_n^{\leq t} = (X_1, Y_1, T_1), ..., (X_{N^t}, Y_{N^t}, T_{N^t})$ be all data up to and including that data point. $f_n^{\leq t}$ is the learning algorithm that consumes only $\mathcal{D}_n^{\leq t}$ and outputs a hypothesis. Denote the generalization error in this setting as $\mathcal{E}^t(f_n^{\leq t}) = \mathbb{E}_{\mathcal{D}}\left[R^t(f_n^{\leq t}(\mathcal{D}_n))\right]$.

Definition 2 (Forward Transfer Efficiency). *The forward transfer efficiency of f_n for task t given n samples is* $\text{FTE}^t(f_n) := \frac{\mathcal{E}^t(f_n^t)}{\mathcal{E}^t(f_n^{\leq t})}$.

In task streaming settings, one can also determine the rate of *backward* transfer by comparing $\mathcal{E}^t(f^{\leq t})$ to the generalization error of the algorithm having sequentially seen the entire training sequence. Backward transfer efficiency is the ratio of (i) the generalization error of the algorithm with access to the data up to and including the last observation from task t, to (ii) the generalization of the learning algorithm with access to the entire data sequence. Thus, this quantity measures the relative effect of future task data on the performance on task t.

Definition 3 (Backward Transfer Efficiency). *The backward transfer efficiency of f_n for task j given n samples is* $\text{BTE}^t(f_n) := \frac{\mathcal{E}^t(f_n^{\leq t})}{\mathcal{E}^t(f_n)}$.

3   Algorithms Our approach to progressive intelligence relies on hypotheses $h : \mathcal{X} \to \mathcal{Y}$ that can be decomposed into three constituent parts: $h(\cdot) = w \circ v \circ u(\cdot)$. The transformer, u, maps an $\mathcal{X}$-valued input into an internal representation space $\tilde{\mathcal{X}}$ [38, 12]. The voter $v : \tilde{\mathcal{X}} \to \mathcal{P}_{Y|\tilde{X}}$ maps the transformed data point into a posterior distribution on the response space $\mathcal{Y}$. Finally, a decider $w : \mathcal{P}_{Y|\tilde{X}} \to \mathcal{Y}$, such as "argmax", produces a predicted label[1]. Our key innovation is realizing that decision rules can ensemble representations learned by transformers *across* tasks. In particular, a representation learned for task t might be a useful representation for task t' and vice versa. Combining these two representations can improve performance on both t and t' and can be extended to an arbitrary number of tasks (Figure 1). Composable hypotheses are more modular and flexible than recursive decompositions, an approach to transfer learning and multi-task learning discussed in [35].

Suppose after n samples we have data from a set of tasks, or environment $\mathcal{T}_n$. We desire algorithms that use data from task t' to transfer knowledge to task t, for all $t, t' \in \mathcal{T}_n$. Let $h_t = w_t \circ v_t \circ u_t$ be the hypothesis function learned for task t. Define the cross-task posterior as the function that votes on classes in class t using the representation output by the transformer for task t'. For example, when using decision trees, this corresponds to learning the partition of a tree from task t', and then pushing data from task t through it to learn the task t vote: $v_t \circ u_{t'}$. Given $|\mathcal{T}_n|$ tasks, there are $|\mathcal{T}_n|$ such cross-task posteriors for each task. The task t decider w_t then combines the votes to obtain a final posterior on $\mathcal{Y}$ learned from all tasks, for example, by averaging. The task-ensembled hypothesis h_t is thus:

$$(1) \qquad\qquad h_t(\cdot) = w_t \circ \frac{1}{|\mathcal{T}_n|} \sum_{t' \in \mathcal{T}_n} v_t \circ u_{t'}(\cdot).$$

In progressive learning settings it is common for the learner to get access to a new task s after learning the set of $\{h_t\}_{t \in \mathcal{T}_n}$. Using the above approach for ensembling representations, incorporating the information from this new dataset is straightforward. Indeed, it only requires learning a single-task composable hypothesis from the new task $h_s = w_s \circ v_s \circ u_s$, the $|\mathcal{T}_n|$ cross-task posteriors $\{v_s \circ u_t\}_{t \in \mathcal{T}_n}$ for task s, and $|\mathcal{T}_n|$ new cross-task posteriors $\{v_t \circ u_s\}_{t \in \mathcal{T}_n}$ for the original $|\mathcal{T}_n|$ tasks. The corresponding functions are updated by augmenting the environment, $\mathcal{T}_n \leftarrow \mathcal{T}_n \cup \{s\}$, and then defining each h_t using Eq. 1. In all cases, we ensure that the posterior outputs are calibrated [23].

3.1   Lifelong Forests A Lifelong Forest (L2F) is a decision forest-based instance of ensembling representations. For each task, the transformer u_t of a L2F is a decision forest [1, 5]. The leaf nodes of each decision tree partition the input space $\mathcal{X}$ [6]. The representation of $x \in \mathcal{X}$ corresponding to a single tree can be a one-hot encoded L_b-dimensional vector with a "1" in the location corresponding

---

[1] In coding theory, these three functions would be called the encoder, channel, and decoder, respectively [10, 9]
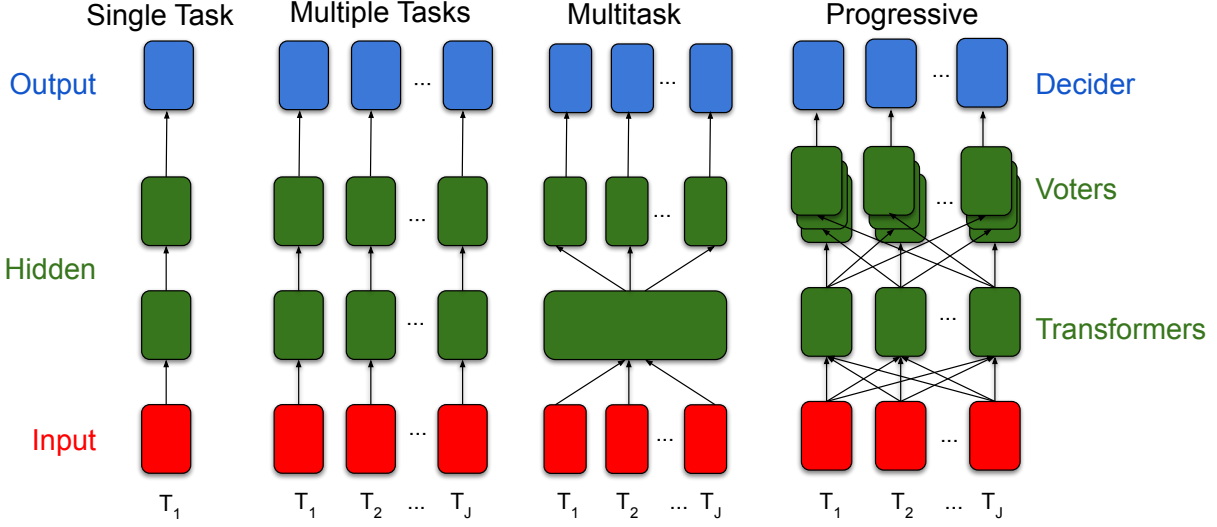
Figure 1: Schemas of composable hypotheses in various learning settings.

to the leaf x falls into of tree b. The representation of x resulting from the collection of trees simply concatenates the B one-hot vectors from the B trees. Thus, the the transformer u_t is the mapping from $\mathcal{X}$ to a B-sparse vector of length $\sum_{b=1}^{B} L_b$. The in-task and cross-task posteriors are learned by populating the cells of the partitions and taking class votes with out-of-bag samples, as in 'honest trees' [6, 11, 2]. The in-task and cross-task posteriors output the average normalized class votes across the collection of trees, adjusted for finite sample bias [23]. The decider w_t averages the in-task and cross-task posterior estimates and outputs argmax to produce a single prediction, as per (1). Recall that honest decision forests are universally consistent classifiers and regressors [2], meaning that with sufficiently large sample sizes, under suitable though general assumptions, they will converge to minimize risk. The single task version of this approaches simplifies to an approach called 'Uncertainty Forests' (UF) [23].

3.2 Lifelong Networks A Lifelong Network (L2N) is a deep network (DN)-based instance of ensembling representations. For each task, the transformer u_t in an L2N is the "backbone" of a DN, including all but the final layer. 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 the DN. In practice, we use a LeNet [18] trained using cross-entropy loss and the Adam optimizer [15] to learn the transformer. In-task and cross-task voters are learned via k-Nearest Neighbors (k-NN) [33]. Recall that a k-NN, with k chosen such that as the number of samples n goes to infinity k goes to infinity and $\frac{k}{n} \to 0$, is a universally consistent classifier [33]. To calibrate the posteriors estimated by k-NN, we rescale them using isotonic regression [26, 29].

4 Simulated Demonstrations of Lifelong Learning Consider a very simple two-task environment: Gaussian XOR and Gaussian Not-XOR (N-XOR), as depicted in Figure 2 (see Appendix A for details). In this environment, the optimal discriminant boundaries are "axis-aligned", and the two tasks share the exact same decision boundary: the coordinate axes. Thus, transferring from one task to the other merely requires learning a bit flip.

Figure 2 shows the generalization error for Lifelong Forest and Uncertainty Forest on *XOR*. L2F and UF achieve the same generalization error on XOR, but UF does not improve its performance on XOR with N-XOR data (because it is a single task algorithm and therefore does not operate on other task data), whereas the performance of L2F continues to improve on XOR given N-XOR data, demonstrating forward transfer. Figure 2 also shows the generalization error for L2F and UF on *N-XOR*. In this case,
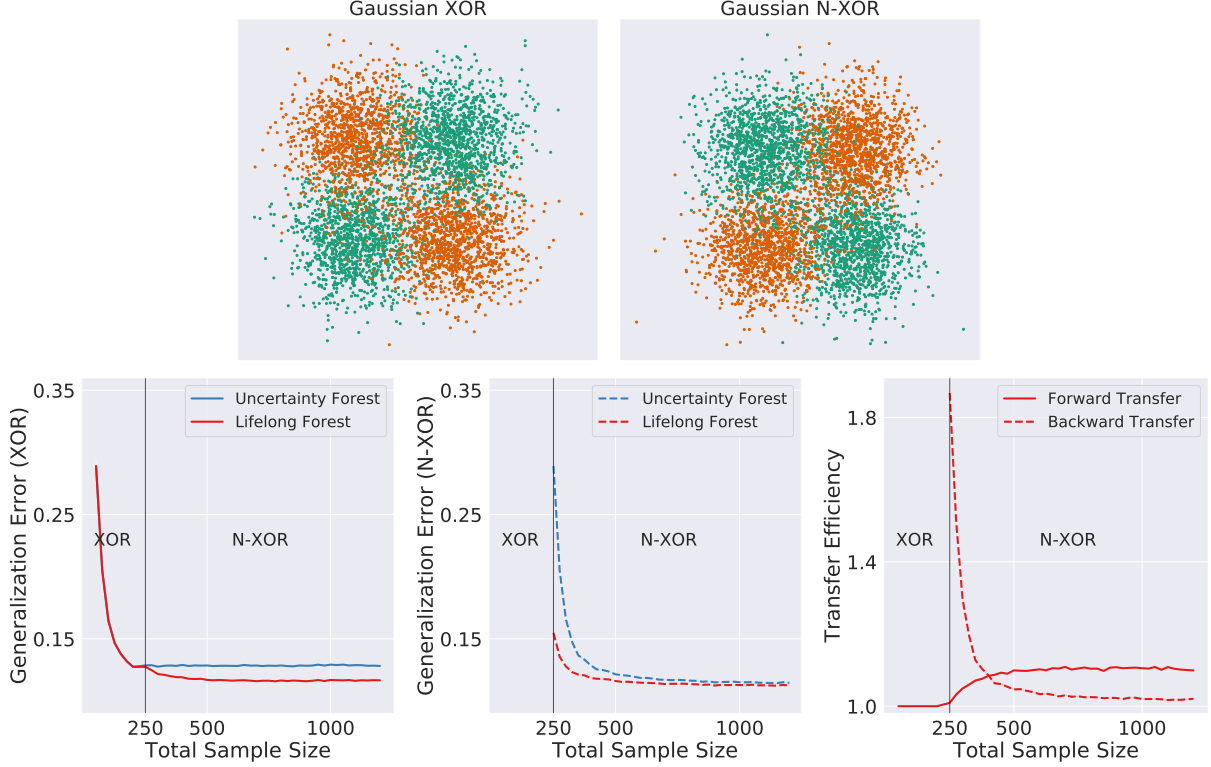
Figure 2: **Lifelong Forests demonstrate forward and backward transfer.** *Top*: 5,000 samples from both Gaussian XOR (left) and N-XOR (right). *Bottom left*: L2F outperforms UF on XOR when N-XOR data is available, demonstrating backward transfer in L2F. *Bottom center*: L2F outperforms UF on N-XOR when prior XOR data is available, demonstrating forward transfer in L2F. *Bottom right*: Transfer Efficiency of L2F. The forward (solid) and backward (dashed) curves are the ratio of the generalization error of L2F to UF in their respective figures. L2F demonstrates both forward and backward transfer in this environment.

UF was trained only on XOR. Both algorithms perform at chance levels until the first N-XOR data arrive. However, L2F improves more rapidly than UF, demonstrating backward transfer.

Finally, Figure 2 shows transfer efficiency for L2F. For XOR (task 1), forward transfer efficiency is one until N-XOR data arrive, and then it quickly ramps up prior to saturating. For N-XOR, backward transfer efficiency of L2F shoots up when N-XOR data arrive, but eventually converges to the same limiting performance of UF. Note that forward transfer efficiency is the ratio of generalization errors for XOR (left panel), and backward transfer efficiency is the ratio of the generalization errors for N-XOR (center panel).

4.1 Lifelong learning in more complex environments

Adversarial Tasks Statistics has a rich history of *robust learning* [14], and machine learning has recently focused on *adversarial learning* [7]. However, in both cases the focus is on adversarial *examples*, rather than adversarial *tasks*. In the context of progressive 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, has no information about 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 Rotated-XOR (R-XOR) (Figure 3, top). Training first on XOR therefore impedes the performance of Lifelong Forests on R-XOR, and thus forward transfer falls below one, demonstrating a graceful forgetting. Because the discriminant boundaries are learned imperfectly with finite data, data from R-XOR can actually improve

performance on XOR, and thus backward transfer is above one.

Spirals Consider an environment with a three spiral and five spiral task (Figure 3, bottom). In this environment, axis-aligned splits are inefficient, because the optimal partitions can be approximated with fewer irregular polytopes than the orthotopes available from axis-aligned splits. The three spiral data helps the five spiral performance because the optimal partitioning for these two tasks is relatively similar to one another, as indicated by forward transfer increasing with increased five spiral data. This is despite the fact that the five spiral task requires more fine partitioning than the three spiral task. Because L2F grows relatively deep trees, it over-partitions space, thereby rendering tasks with more coarse optimal decision boundaries useful for tasks with more fine optimal decision boundaries. The five spiral data also improves the three spiral performance, as long as there are a sufficient number of samples from the five spiral task to adequately estimate the posteriors within each cell.

5   Real data experiments

5.1   Vision Benchmarks The CIFAR 100 challenge [17], consists of 50,000 training and 10,000 test samples, each a 32x32 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 [19] (see Appendix C for details). We compare Lifelong Forests and Lifelong Networks, to several state of the art deep learning based lifelong learning algorithms, including Deconvolution-Factorized CNNs (DF-CNN) [19], elastic weight consolidation (EWC) [16], Online EWC [32], Synaptic Intelligence (SI) [39], Learning without Forgetting (LwF) [20], and Progressive Neural Networks (Prog-NN) [30]. Figure 4 shows the forward, backward, and overall transfer efficiency for each algorithm on this benchmark dataset.
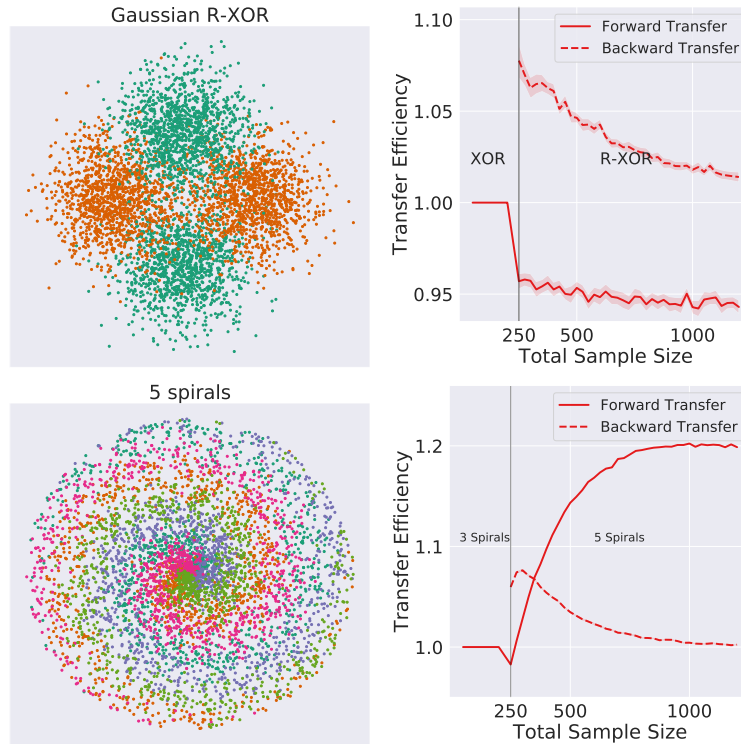

Figure 3: **Lifelong Forests forward and backward transfer capabilities in more complex environments (including an adversarial task)**. *Top*: R-XOR has orthogonal decision boundaries to XOR. Nonetheless, Lifelong Forests shows positive forward transfer efficiency and graceful forgetting. *Bottom*: Spiral with three classes and spiral with five classes (only five classes shown). Although the five class problem has a finer optimal partition, L2F can transfer knowledge both forward and backward.
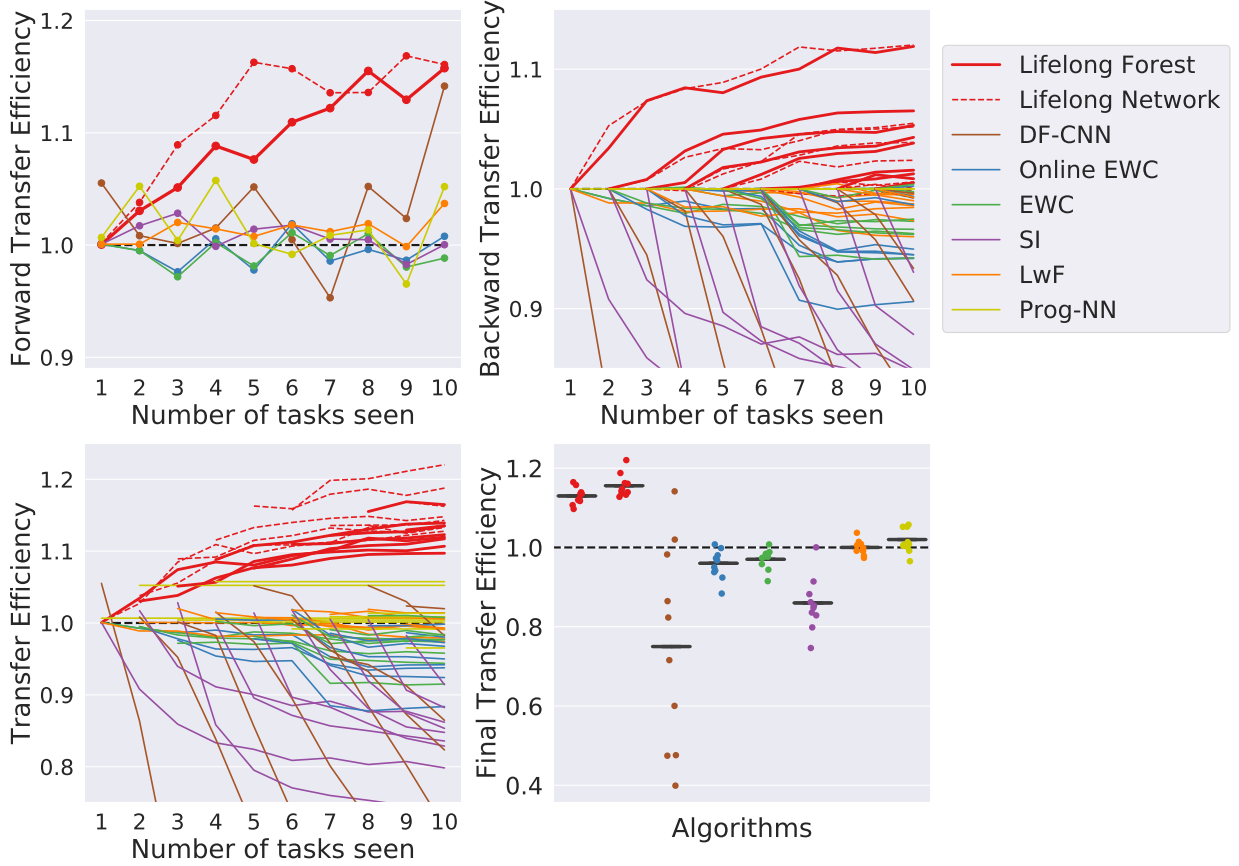
Figure 4: **Performance of different algorithms on the CIFAR 10x10 vision experiments.** *Top left*: Lifelong Forests (L2F) and Lifelong Networks (L2N) consistently demonstrate forward transfer for each task. *Top right*: L2F and L2N uniquely exhibit backward transfer. *Bottom left*: Transfer efficiency for L2F and L2N uniquely improve as more tasks are seen. *Bottom right*: Scatter plot of final transfer efficiency of 10 tasks for each algorithm (mean shown by horizontal black lines). Only L2F and L2N demonstrate transfer efficiency for all tasks above 1.

Forward transfer efficiency of Lifelong Forests and Lifelong Networks are (approximately) monotonically increasing. This indicates that as they observe data from new tasks, they were able to leverage the data from previous tasks to improve performance on the new tasks. This is in contrast to the other approaches, none of which appear to reliably demonstrate any degree of forward transfer (top left). Similarly, the backward transfer efficiencies of Lifelong Forests and Lifelong Networks are (approximately) monotonically increasing for each task. This indicates that as they observe data from new tasks, they are able to *improve* performance on previous tasks, relative to their performance on those tasks prior to observing new out-of-task data. That is, L2F and L2N progressively learn. This is in contrast to the other approaches, none of which appear to demonstrate any degree of backward transfer. To the contrary, the previously proposed approaches appear to demonstrate catastrophic forgetting or do not transfer at all (top right). Taken together, transfer efficiency for L2F and L2N are (approximately) monotonically increasing, yielding a $10\% - 20\%$ improvement on most tasks by virtue of other task data; in contrast, most other approaches fail to demonstrate transfer for the majority of tasks (bottom).

5.2 Adversarial and Rotated Experiments Consider the same CIFAR 10x10 experiment above, but, for tasks 2 through 9, randomly permuted the class labels within each task, rendering each of those tasks adversarial with regard to the 1st task. Figure 5 (left) indicates that LF gracefully forgets
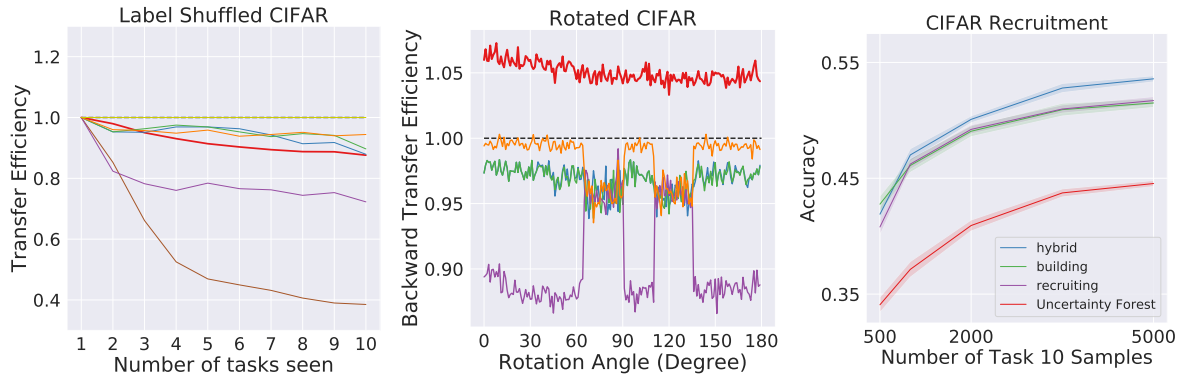
Figure 5: **Extended CIFAR 10x10 experiments.** *Bottom left*: Shuffling class labels within tasks two through nine demonstrates that while LF utilizes this information, previous approaches seem to be invariant to task labels. *Bottom center*: LF is nearly invariant to rotations, whereas other approaches fail to transfer even when the data from a second task is sampled from an identical distribution as the first task. *Bottom Right*: Building and recruiting ensembles are two boundaries of a continuum. The hybrid approach outperforms both building and re-allocating resources in terms of accuracy on the tenth task.

in the presence of these nine adversarial tasks. The other algorithms seem invariant to label shuffling (comparing performance with Figure 4 indicates that performance is effectively unchanged), suggesting that they fail to take advantage of class labels for transferring to other tasks.

Now, consider a Rotated CIFAR experiment, which uses only data from the 1st task, divided into two equally sized subsets (making two tasks), where the second subset is rotated. Figure 5 (center) shows that L2F's transfer efficiency is nearly invariant to rotation angle, whereas approaches fail to transfer for any angle. Note that zero rotation angle corresponds to the two tasks *having identical distributions*; the fact that none of the other algorithms transfer even in this setting suggests that they cannot transfer at all. Appendix C contains additional experiments with repeated tasks or subtasks, demonstrating similar results: L2F and L2N transfer efficiently in these contexts as well.

5.3 From building new, to recruiting existing, resources The lifelong learning approaches presented so far either build new, or recruit existing, resources (L2F and L2N both exclusively build resources). However, this binary distinction is unnecessary and unnatural: in biological learning, systems develop from building to recruiting resources. Figure 5 (right) demonstrates that variants of L2F can span the continuum from purely building to purely recruiting resources. We trained L2F on the first nine CIFAR 10x10 tasks using 50 trees per task. For the tenth task, we could select the 50 trees (out of the 450 existing trees) that perform best on task 10 (recruiting). Or, we could train 50 new trees, as L2F would normally do (building). Or we do a hybrid: building 25 and recruiting 25 (hybrid). This hybrid approach performs better than either alternative, suggesting that one can improve our representation ensembling approach by dynamically determining whether to build a new representation, and if so, how to optimally combine those resources with existing resources.

5.4 Language Benchmarks Neither L2F nor L2N leverage any modality specific architecture, such as convolutions. This allows them to be applicable to other modalities out-of-the-box. We consider two natural language processing environments, both using 8 million sentences downloaded from Wikipedia, and trained a 16 dimensional Fasttext [4] embedding of tokenized words and 2-4 character n-grams from these sentences. These embeddings served as the input to our ensembling representation algorithms (see Appendix D for details). In the first experiment, the goal was to identify the language associated with a given sentence. In the second experiment, the goal was to identify which of 20 different Bing entity types best matches a word or phrase. For example, the entity type for "Johns Hopkins

University" is "education.school". Appendix Figure 4 shows L2F demonstrates backwards transfer in both of these environments.

6 Discussion We introduced representation ensembling as a general approach to progressive learning. The two specific algorithms we developed demonstrate the possibility of achieving both forward and backward transfer. Previous approaches relied on either (i) always building new resources, or (ii) only recruiting existing resources, when confronted with new tasks. We demonstrate that a hybrid approach outperforms both edge cases, motivating the development of more elaborate dynamic systems.

We hope this work motivates a tighter connection between biological and machine learning. By carefully designing behavioral and cognitive human and/or non-human animal experiments, in tandem with neurophysiology, gene expression, anatomy and connectivity, we may be able to infer more about how neural systems are able to progressively learn so efficiently. By designing such experiments in neurodevelopment, neuro-impairment, and neurodegeneration, may yield valuable information regarding limiting or reversing memory or learning impairments.

References

[1] Yali Amit and Donald Geman. "Shape Quantization and Recognition with Randomized Trees". In: Neural Comput. 9.7 (Oct. 1997), pp. 1545–1588.

[2] S. Athey, J. Tibshirani, and S. Wager. "Generalized random forests". In: Annals of Statistics 47.2 (2019), pp. 1148–1178.

[3] Peter J Bickel and Kjell A Doksum. Mathematical statistics: basic ideas and selected topics, volumes I-II package. Chapman and Hall/CRC, 2015.

[4] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: arXiv preprint arXiv:1607.04606 (2016).

[5] Leo Breiman. "Random forests". In: Machine learning 45.1 (2001), pp. 5–32.

[6] Leo Breiman et al. Classification and regression trees. CRC press, 1984.

[7] Joan Bruna et al. "Intriguing properties of neural networks". Dec. 2013.

[8] Rich Caruana. "Multitask learning". In: Machine learning 28.1 (1997), pp. 41–75.

[9] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: (June 2014). arXiv: 1406.1078 [cs.CL].

[10] Thomas M Cover and Joy A Thomas. Elements of Information Theory. en. New York: John Wiley & Sons, Nov. 2012.

[11] M. Denil, D. Matheson, and N. De Freitas. "Narrowing the Gap: Random Forests In Theory and In Practice". In: Proceedings of the 31st International Conference on Machine Learning. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research. June 2014, pp. 665–673.

[12] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (Oct. 2018). arXiv: 1810.04805 [cs.CL].

[13] Chelsea Finn et al. "Online Meta-Learning". In: (Feb. 2019). arXiv: 1902.08438 [cs.LG].

[14] Peter J Huber. Robust statistical procedures. Vol. 68. Siam, 1996.

[15] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: arXiv preprint arXiv:1412.6980 (2014).

[16] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: Proceedings of the national academy of sciences 114.13 (2017), pp. 3521–3526.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[18] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: Proceedings of the IEEE 86.11 (1998), pp. 2278–2324.

[19] Seungwon Lee, James Stokes, and Eric 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.

[20] Zhizhong Li and Derek Hoiem. "Learning without forgetting". In: IEEE transactions on pattern analysis and machine intelligence 40.12 (2017), pp. 2935–2947.

[21] James L McClelland, Bruce L McNaughton, and Randall 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." In: Psychological review 102.3 (1995), p. 419.

[22] Michael McCloskey and Neal J Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: Psychology of learning and motivation. Vol. 24. Elsevier, 1989, pp. 109–165.

[23] Ronak Mehta et al. "Estimating Information-Theoretic Quantities with Random Forests". In: arXiv preprint arXiv:1907.00325 (2019).

[24] Tom M Mitchell. "Machine learning and data mining". In: Communications of the ACM 42.11 (1999), pp. 30–36.

[25] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of Machine Learning. en. MIT Press, Nov. 2018. URL: https://market.android.com/details?id=book-dWB9DwAAQBAJ.

[26] Alexandru Niculescu-Mizil and Rich Caruana. "Predicting good probabilities with supervised learning". In: Proceedings of the 22nd international conference on Machine learning. ICML '05. Bonn, Germany: Association for Computing Machinery, Aug. 2005, pp. 625–632.

[27] German I Parisi et al. "Continual lifelong learning with neural networks: A review". In: Neural Networks (2019).

[28] German I. Parisi et al. "Continual lifelong learning with neural networks: A review". In: Neural Networks 113 (2019), pp. 54–71. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2019.01.012. URL: http://www.sciencedirect.com/science/article/pii/S0893608019300231.

[29] Vickram Rajendran and William LeVine. "Accurate Layerwise Interpretable Competence Estimation". In: Advances in Neural Information Processing Systems. 2019, pp. 13981–13991.

[30] Andrei A Rusu et al. "Progressive neural networks". In: arXiv preprint arXiv:1606.04671 (2016).

[31] Paul Ruvolo and Eric Eaton. "ELLA: An efficient lifelong learning algorithm". In: International Conference on Machine Learning. 2013, pp. 507–515.

[32] Jonathan Schwarz et al. "Progress & compress: A scalable framework for continual learning". In: arXiv preprint arXiv:1805.06370 (2018).

[33] Charles J Stone. "Consistent Nonparametric Regression". en. In: Ann. Stat. 5.4 (July 1977), pp. 595–620.

[34] Sebastian Thrun. "Is learning the n-th thing any easier than learning the first?" In: Advances in neural information processing systems. 1996, pp. 640–646.

[35] Sebastian Thrun and Lorien Pratt. Learning to Learn. en. Springer Science & Business Media, Dec. 2012. URL: https://market.android.com/details?id=book-X_jpBwAAQBAJ.

[36] L G Valiant. "A Theory of the Learnable". In: Commun. ACM 27.11 (Nov. 1984), pp. 1134–1142. URL: http://doi.acm.org/10.1145/1968.1972.

[37] V Vapnik and A Chervonenkis. "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities". In: Theory Probab. Appl. 16.2 (Jan. 1971), pp. 264–280.

[38] Ashish Vaswani et al. "Attention is All you Need". In: Advances in Neural Information Processing Systems 30. Ed. by I Guyon et al. Curran Associates, Inc., 2017, pp. 5998–6008.

[39] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence". In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org. 2017, pp. 3987–3995.

[40] Jing Zhao et al. "Comparing Bilingual to Monolingual Learners on English Spelling: A Meta-analytic Review". en. In: Dyslexia 22.3 (Aug. 2016), pp. 193–213.
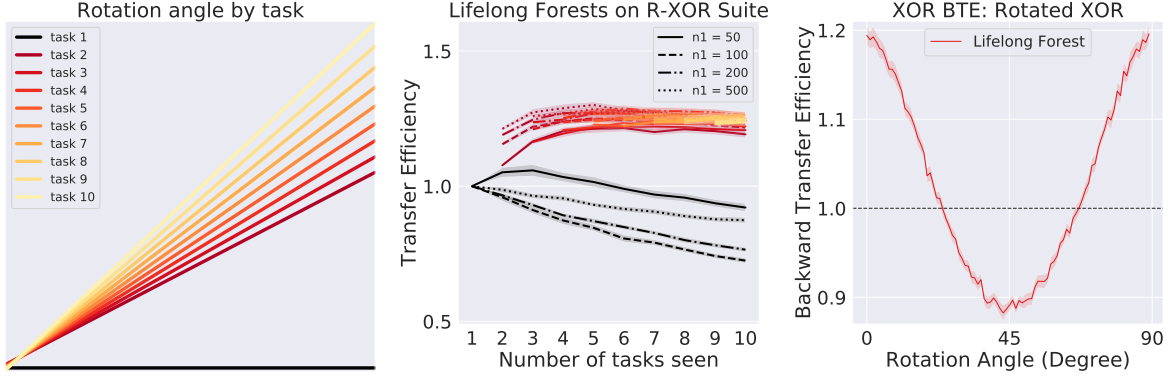
Figure 1: Lifelong Forests transfers knowledge across similar tasks and gracefully forgets when presented with adversarial tasks. *Left*: Rotated XOR tasks: the angle between the origin and the line is the rotation angle of the XOR problem. *Middle*: Transfer efficiency of Lifelong Forest. Linestyles are used to differentiate the amount of Task 1 (XOR) data available. All other tasks had 100 training samples. LF transfers when tasks are sufficiently similar and/or sample sizes are chose appropriately, and gracefully forgets otherwise. *Right*: Backward transfer efficiency for XOR as a function of rotation angle. Rotations far from $45°$ result in helpful transfer.

Appendix A. Simulated Results.

In the following simulation, we construct an environment with two tasks. For each, we sample 250 times from the first task, followed by 750 times from the second task. These 1,000 samples comprise the training data. We sample another 1,000 hold out samples to evaluate the algorithms. We fit an Uncertainty Forest (UF) (an honest forest with a finite-sample correction [3]) for both tasks and use the learned trees to subsequently fit a Lifelong Forest. We repeat this process 1,500 times to obtain errorbars. Error bars correspond to the 95% confidence intervals.

A.1 Gaussian XOR
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 $\pm \begin{bmatrix} 0.5, & 0.5 \end{bmatrix}^T$, 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 $\pm \begin{bmatrix} 0.5, & -0.5 \end{bmatrix}^T$, and variances proportional to the identity matrix. Gaussian N-XOR is the same distribution as Gaussian XOR with the class labels flipped. Rotated XOR (R-XOR) rotates XOR by $\theta°$ degrees.

A.2 Rotated Gaussian XOR Suite
We study Lifelong Forests on a sequence of 10 rotated Gaussian XOR tasks. Representations of the tasks are shown on the left of Figure 1. The angle between the line and the origin corresponds to the rotation angle for that task. In particular, the line $y = 0$ for Task 1 corresponds to Gaussian XOR. The line $y = x$ for Task 10 corresponds to Gaussian $45°$-XOR. The eight other tasks were chosen to be progressively less similar to Gaussian XOR. This set of tasks was chosen to study both the effect of many tasks that are dissimilar to a particular task (through the performance of Lifelong Forests on Task 1, Gaussian XOR) and the effect of many tasks close to a particular task (through the performance of Lifelong Forests on Tasks 2-10). For the suite of rotated XORs, a pair of tasks with a small difference in rotation angle between them will result in helpful transfer.

Figure 1 (center) shows the transfer efficiency of Lifelong Forests on the rotated XOR suite. Tasks

Table 1: Task splits for CIFAR 10x10.

Task #	Image Classes
1	apple, aquarium fish, baby, bear, beaver, bed, bee, beetle, bicycle, bottle
2	bowl, boy, bridge, bus, butterfly, camel, can, castle, caterpillar
3	chair, chimpanzee, clock, cloud, cockroach, couch, crab, crocodile, cup, dinosaur
4	dolphin, elephant, flatfish, forest, fox, girl, hamster, house, kangaroo, keyboard
5	lamp, lawn mower, leopard, lion, lizard, lobster, man, maple tree, motor cycle, mountain
6	mouse, mushroom, oak tree, orange, orchid, otter, palm tree, pear, pickup truck, pine tree
7	plain, plate, poppy, porcupine, possum, rabbit, raccoon, ray, road, rocket
8	rose, sea, seal, shark, shrew, skunk, skyscraper, snail, snke, spider
9	squirrel, streetcar, sunflower, sweet pepper, table, tank, telephone, television, tiger, tractor
10	train, trout, tulip, turtle, wardrobe, whale, willow tree, wolf, woman, worm

2 through 9 had 100 training samples per task. Task 1 had a variable amount of training data. When the number of XOR training samples is small, the representations learned from the tasks close to XOR help improve performance. When the number of training samples from XOR is sufficiently large that the representation learned for XOR is nearly optimal, other tasks cause a graceful forgetting. For an intermediate number of XOR training samples, other task data causes greater forgetting. The representations from the non-XOR tasks are mutually beneficial, as indicated by the (generally) upward trend of the transfer efficiency for each task.

Finally, consider a two task environment, where Task 1 is XOR, and Task 2 is θ-XOR, for differing values of θ. Figure 1 shows that θ-XOR can improve performance on XOR whenever the rotation angle is far from $45°$.

Appendix B. Neural Network Architectures.

We compared our approaches to six reference lifelong learning methods. These algorithms can be classified into two groups based on whether they build or recruit resources given new tasks. Among them, Prog-NN [4] and DF-CNN [2] learn new tasks by building new resources. The other four algorithms, EWC, Online-EWC, SI and LwF, recruit existing resources. EWC, Online-EWC and SI rely on preferentially updating the network parameters depending on their relative importance to the previous task. On the other hand, LwF predicts the labels of the input data from the current task using the model trained on the previous tasks. These predicted labels act as the soft targets for the current training data, i.e., (input data, soft target) pairs are used in the regularization term with the (input data, original target) pairs being used in the main loss function. This prevents the deviation of the parameters too much from the optimum value for the previous tasks while at the same time, it enables the network to learn a new task. The implementations for all of the algorithms are adapted from the codes provided by the authors of [2] and [5]. The codes are modified to work on the CIFAR 10x10 setting without any change in the parameters. All algorithms used the default hyperparameters.

Appendix C. CIFAR 10x10 Extended Results.

C.1   CIFAR 10x10   Table 1 shows the image classes associated with each task number. Table 2 shows a set of summary statistics from the CIFAR 10x10 experiment of Section 5.1 of the main text. Notably, only Lifelong Forests and Lifelong Networks have both mean and minimum transfer efficiency greater than 1. Further, Table 3 shows the final transfer efficiencies for each algorithm studied for convenience and reproducibility.

C.2   Repeated Classes   We also considered the setting where each task is defined by a random sampling of 10 out of 100 classes with replacement. This environment is designed to demonstrate the effect of tasks with shared subtasks, which is a common property of real world learning tasks. This

Table 2: Summary of cross transfer of learning between tasks for different algorithms.

Algorithm	Mean TE	Min. TE	Mean FTE (Task 10)	Mean BTE(Task 1)
L2F	**1.13**(± 0.01)	**1.10**(± 0.01)	**1.16**(± 0.01)	**1.12**(± 0.01)
L2N	**1.16**(± 0.01)	**1.13**(± 0.02)	**1.16**(± 0.02)	**1.12**(± 0.01)
DF-CNN	0.75(± 0.08)	0.40(± 0.01)	1.14(± 0.01)	0.38(± 0.01)
Online EWC	0.96(± 0.01)	0.88(± 0.01)	1.01(± 0.01)	0.94(± 0.02)
EWC	0.97(± 0.01)	0.91(± 0.01)	0.99(± 0.01)	0.97(± 0.01)
SI	0.86(± 0.02)	0.75(± 0.01)	1.00(± 0.01)	0.80(± 0.01)
LwF	1.00(± 0.01)	0.97(± 0.01)	1.04(± 0.01)	0.97(± 0.01)
ProgNN	1.02(± 0.01)	0.97(± 0.01)	1.05(± 0.01)	1.00(± 0.00)

Table 3: Final transfer efficiency on each task for different algorithms.

Algorithm	Task									
	1	2	3	4	5	6	7	8	9	10
L2F	**1.12**	**1.10**	**1.11**	**1.14**	**1.12**	**1.12**	**1.14**	**1.16**	**1.14**	**1.16**
L2N	**1.14**	**1.13**	**1.14**	**1.15**	**1.19**	**1.22**	**1.14**	**1.13**	**1.16**	**1.16**
DF-CNN	0.40	0.47	0.48	0.60	0.72	0.82	0.86	0.98	1.02	1.14
Online EWC	0.94	0.94	0.88	0.95	0.92	0.97	0.97	1.00	0.98	1.01
EWC	0.97	0.96	0.91	0.98	0.94	0.98	0.98	1.01	0.98	0.99
SI	0.80	0.86	0.75	0.84	0.83	0.85	0.85	0.88	0.91	1.00
LwF	0.97	0.99	0.98	1.00	0.99	1.01	1.01	1.01	0.99	1.04
ProgNN	1.01	1.05	1.00	1.06	1.00	0.99	1.01	1.01	0.97	1.05

setting generalizes the previously proposed "Class-Incremental" and "Task-Incremental" distinction [5]. Figure 2 shows transfer efficiency of Lifelong Forests on Task 1. Finally, we considered a curriculum where tasks repeat, in contrast to CIFAR 10x10. Figure 3 demonstrates that L2F consistently transfers in this setting as well.

Appendix D. Language Experiments.

D.1 Language Detection 10x3 We downloaded a language identification corpus consisting of around 8 million sentences and 350 languages from `https://tatoeba.org/eng/downloads`, and trained a 16 dimensional Fasttext [1] embedding of tokenized words and 2-4 character n-grams from these sentences using a character based skip-gram model without using the language labels. We then picked 30 languages and randomly chose 150 sentences for training and 2500 sentences for testing for every language but Bosnian. For Bosnian we used 150 sentences for training and 396 sentences for testing due to a limited number of samples. A sentence embedding is found by averaging all L2-normalized word and n-grams embedding vectors within a sentence.

We split the 30 languages into ten 3 class tasks and tasks are presented one at a time. Class splits are given in Table 4. The backward transfer efficiencies of Lifelong Forests for the ten tasks are shown
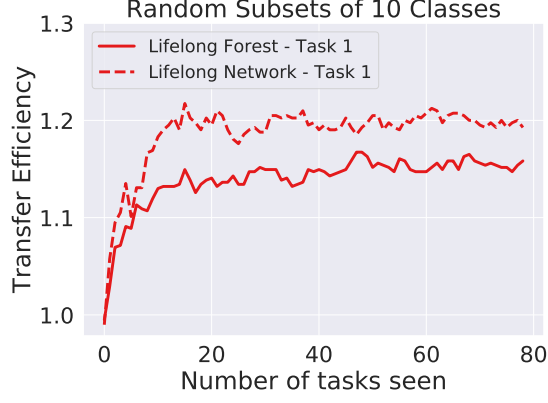
Figure 2: Lifelong Forests transfer knowledge effectively when tasks share common classes. Each task is a random selection of 10 out of the 100 CIFAR-100 classes. Both L2F and L2N demonstrate transfer efficiency of $10\% - 20\%$ for Task 1.
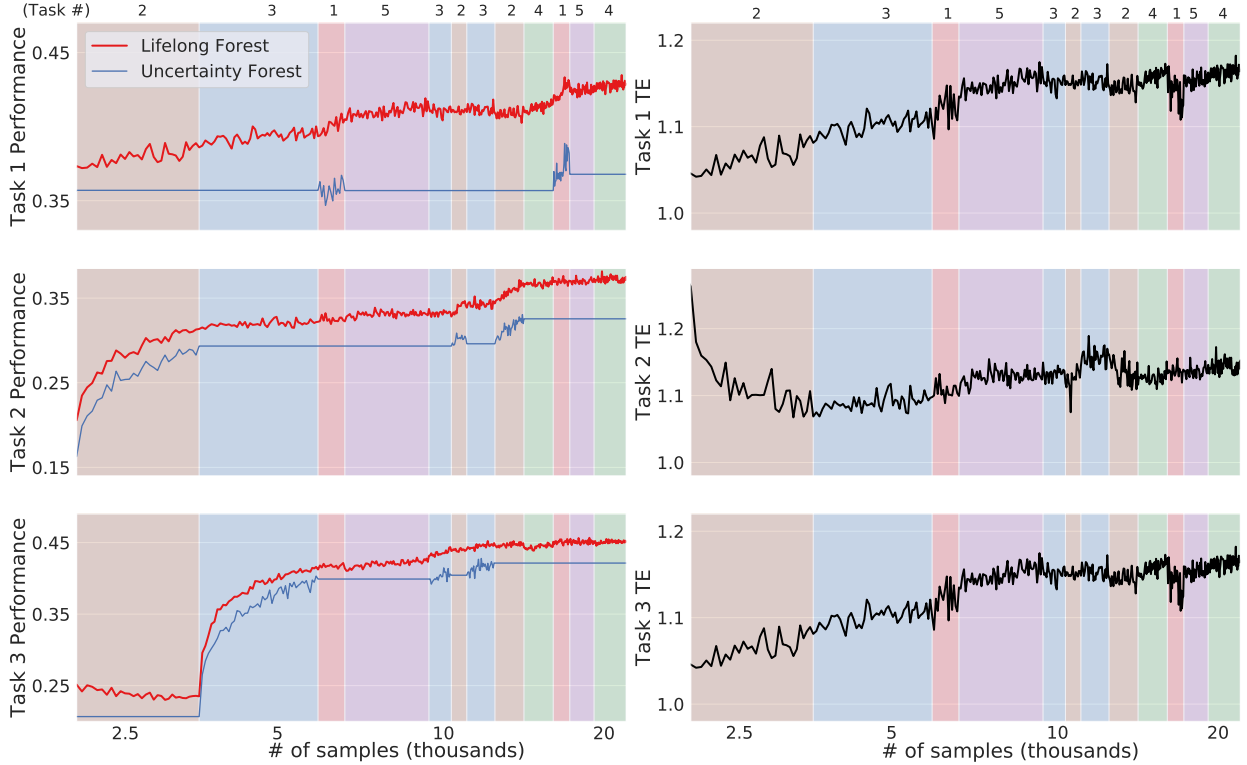

Figure 3: **Performance of lifelong forest on CIFAR-100.** Task 1, 2, and 3 are shown in the top, middle, and bottom, respectively. *Left*: L2F's performance (1-generalization error) improves with additional data from any of the five considered tasks. UF, by definition, only improves when it sees the relevant task data. *Right*: Transfer efficiency (ratio of the L2F to UF). x-axis is on a log scale.

in the left panel of Figure 4. Lifelong Forests, generally, transfer knowledge across the stream of tasks.

D.2 Entity Type Detection 5x4 An entity type is a label of an entity, such as "Johns Hopkins University", that provides a description of the entity, such as "education.school". We obtained a proprietary entity name and type table from Bing catalogs. For each entity we generated an embedding using a pre-trained English Fasttext of 1 million-word vectors trained on Wikipedia 2017, UMBC webbase

Table 4: Task splits for language detection 10x3.

Task Number	Language Classes
1	Swedish, Norwegian BokmÃ̆l, Danish
2	Mandarin Chinese, Yue Chinese, Wu Chinese
3	Russian, Ukrainian, Polish
4	Spanish, Italian, Portuguese
5	Finnish, Hungarian, Estonian
6	English, Dutch, German
7	Croatian, Serbian, Bosnian
8	Japanese, Korean, Vietnamese
9	Hebrew, Arabic, Hindi
10	French, Catalan, Breton

Table 5: Class splits for Entity Type Detection 5x4.

Task Number	Entity Type Classes
1	american_football.player, biology.organism_classification, book.author, book.book
2	book.edition, book.written_work, business.operation, commerce.consumer_product
3	education.field_of_study, education.school, film.actor, film.character
4	film.film, media_common.actor, music.artist, music.group
5	organization.organization, people.person, tv.series_episode, tv.program

corpus and statmt.org news datasets (16 billion tokens). The entity name embedding used was the summation of the L2-normalized vectors of for all tokens corresponding to the entity name.

We took the entity name embedding vectors for 20 entity types. For each type, we used 10,000 entity names for training and 1,000 entity names as a testing set. That is, we classified entity types based on their names. We split the 20 entity types into 5 tasks of 4 classes each. Tasks are presented one at a time. Entity types and task splits are given in Table 5. The backward transfer efficiencies corresponding to each task are shown in the right panel of Figure 4. Again, Lifelong Forests transfer knowledge across the stream of tasks.
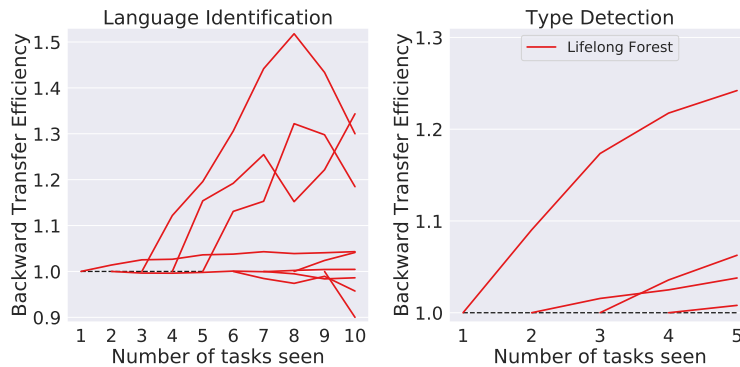

Figure 4: Lifelong Forests transfer knowledge across multiple tasks in two natural language settings.

References

[1] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: arXiv preprint arXiv:1607.04606 (2016).

[2] Seungwon Lee, James Stokes, and Eric 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.

[3] Ronak Mehta et al. "Estimating Information-Theoretic Quantities with Random Forests". In: arXiv preprint arXiv:1907.00325 (2019).

[4] Andrei A Rusu et al. "Progressive neural networks". In: arXiv preprint arXiv:1606.04671 (2016).

[5] Gido M van de Ven and Andreas S Tolias. "Three scenarios for continual learning". In: (Apr. 2019). arXiv: 1904.07734 [cs.LG].