Tackling Non-forgetting and Forward Transfer with a Unified Lifelong Learning Approach

Xinyu Yun [*1] **Tanner Bohn** [*1] **Charles X. Ling** [1]

Abstract

Humans are the best example of agents that can learn a variety of skills incrementally over the course of their lives, and imbuing machines with this skill is the goal of lifelong machine learning. Ideally, lifelong learning should achieve non-forgetting, forward and backward transfer, avoid confusion,support few-shot learning, and so on. In previous approaches, the focus has been given to subsets of these properties, often by fitting together an array of separate mechanisms. In this work, we propose a simple yet powerful unified framework that supports almost all of these properties through *one* central consolidation mechanism. We then describe a particular instance of this framework designed to support non-forgetting and forward transfer. This novel approach works by efficiently locating sparse neural sub-networks and controlling their consolidation during lifelong learning.

1. Introduction

The past decade has seen significant growth in the capabilities of artificial intelligence. Deep learning in particular has achieved great successes in medical image recognition and diagnostics (Litjens et al., 2017; Shen et al., 2017), tasks on natural language processing (Radford et al., 2019; Devlin et al., 2019), difficult games (Silver et al., 2017), and even farming (Kamilaris & Prenafeta-Boldú, 2018). However, deep learning models almost always need thousands or millions of training samples to perform well. This is in a sharp contrast with human learning, which normally learns a new concept with a small number of samples. Two other major weaknesses in current deep learning when compared to human learning, as conveyed in Figure 1, include difficulty in learning many tasks sequentially without forgetting previous ones and leveraging previous learned knowledge to better acquire new knowledge.

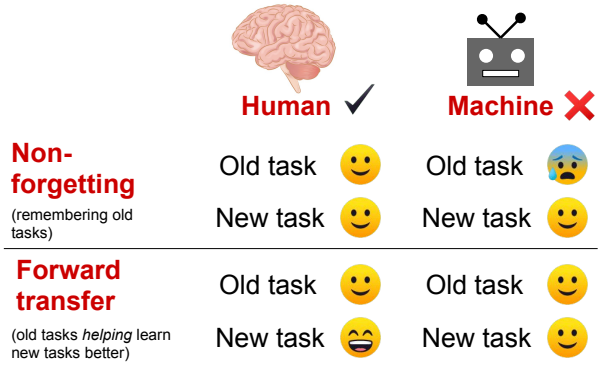

Figure 1. Illustrations of two key lifelong learning properties of non-forgetting and forward transfer. Humans are generally able to learn new tasks without forgetting old ones and use old tasks to better learn new ones. However, machine learning is susceptible to forgetting previous tasks and being unable to transfer knowledge to new tasks. Even more learning properties are discussed in Section 2.2. Image of brain sourced from `https://www.injurymap.com/free-human-anatomy-illustrations` under CC license.

Several lines of research in supervised learning exist to overcome these weaknesses. Multi-task learning (Caruana, 1997) considers how to learn multiple concepts at the same time such that they help each other to be learned better. The related field of transfer learning (Pan & Yang, 2009) assumes that some concepts have been previously learned and we would like to transfer their knowledge to assist learning new concepts. Few-shot learning (Fei-Fei et al., 2006) aims to learn tasks with a small number of labeled data. Lifelong learning (Thrun, 1998; Thrun., 1995) (LLL – also known as continual (Parisi et al., 2019) or sequential learning (McCloskey & Cohen, 1989)) considers how to learn and transfer skills across long sequences of tasks.

However, most previous approaches can only demonstrate

[*]Equal contribution [1]Computer Science, Western University, London, ON, Canada. Correspondence to: Xinyu Yun <xyun@uwo.ca>, Tanner Bohn <tbohn@uwo.ca>, Charles X. Ling <charles.ling@uwo.csd.ca>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

subsets of these human-like properties, often by different complex mechanisms. For example, existing lifelong learning (LLL) techniques tend to use one or more of three types of mechanisms, each of which comes with their own drawbacks and hurdles (De Lange et al., 2019). These mechanisms are based on replay, regularization, and dynamic architecture respectively. See Section 2.3 for overviews of these mechanisms. We wish to find a learning framework with one central mechanism that can seamlessly implement many desirable properties.

In this paper, we propose a unified approach that brings lifelong machine learning closer to human learning by supporting many desirable properties. This unified framework, described in Section 3, places weight consolidation as the central mechanism to achieve many lifelong learning properties. A more extensive description of this framework is provided in (Ling & Bohn, 2019). Starting from this unified framework, we then describe a specific novel approach to LLL. This approach works by identifying the sub-networks critical to performance on each learned class (inspired by the lottery ticket hypothesis (Frankle & Carbin, 2018)) and intelligently using this information to inform weight consolidation policies.

Our primary contributions are:

- We describe a conceptual unified LLL framework which is designed to display many desirable learning properties with a single central mechanism (see Section 3).
- We describe a particular lifelong learning approach which works by identifying sub-networks critical to individual classes in order to intelligently control weight consolidation (see Section 4).
- We demonstrate that by changing a small number of hyperparameters, our approach can achieve non-forgetting and support forward transfer (see Section 5).

2. Background

In this section we will first describe our lifelong learning (LLL) setting. Second, we will discuss a broad set of important LLL properties. Third, we will provide an overview of common mechanisms employed by LLL approaches.

2.1. Lifelong Learning Setting

In our lifelong setting, we consider the *task-incremental* classification tasks, where batches of data for new tasks arrive sequentially. That is, a sequence of $(T_1, D_1), (T_2, D_2), ...$ are given, where D_i is the labeled training data of task T_i (from the space of tasks $\mathcal{T}$), and an individual task consists of a set of classes to be learned. Classification models for $(T_1, T_2, ..., (T_k$ must be functional before (T_{k+1}, D_{k+1}) arrives. This models the incremental process of human lifelong learning.

This LLL setting can be easily extended to other learning settings such as multi-task learning where all T_i and D_i are given at the same time. It can also be extended to meta-learning where the knowledge learned for T_i only provides a starting point to learn T_j more effectively. Additionally, it can be extended to curriculum learning where T_i are able to be ordered from "simple" to "complex" based on a given difficulty metric.

2.2. Lifelong Learning Properties

Here we discuss at a high level several properties a LLL approach would ideally exhibit. In this particular work, we design an approach with the first three properties in mind.

Continual learning and testing: Before starting to learn a new task T_j, a LLL approach should be able to perform well on all $T_{i<j}$. While learning the new task T_j, LLL should *only* use data D_j. This is a startling contrast with the standard multi-task (batch) learning where data of all tasks are used for training at the same time. This ensures that the model will be consistently useful when continually learning tasks with high computational and data efficiency. See Section 3.2 for details.

Non-forgetting: This is the ability to avoid catastrophic forgetting (McCloskey & Cohen, 1989), where learning T_j causes a dramatic loss in performance on one or more $T_{i<j}$. Ideally, learning T_j using only the data of T_j would not affect $T_{i<j}$. Due to the tendency towards catastrophic forgetting, non-lifelong learning approaches would require retraining on data for all tasks together to avoid forgetting. This may reduce computational and data efficiency. A property opposite to non-forgetting is **graceful forgetting** (Aljundi et al., 2018), as often seen in humans. Learning new tasks may require additional model capacity, and when this is not possible, the model can perform graceful forgetting of unimportant tasks to free up capacity for new tasks. See Section 3.3 for details.

Forward transfer: This is the ability to learn new tasks, $T_{\geq i}$, easier and better following earlier learned tasks $T_{<i}$, also known as knowledge transfer (Pan & Yang, 2009). See Section 3.4 for details.

Few-shot learning: Achieving sufficient forward transfer opens the door to few-shot learning of later concepts.

Backward transfer: This is knowledge transfer from $T_{\geq i}$ to $T_{<i}$ – the opposite direction as forward transfer. When learning new tasks T_j they may in turn help to improve the performance of $T_{i<j}$. This is like an "overall review" before a final exam after materials of all chapters have been

taught and learned. Later materials can often help better understand earlier materials.

Adapting to concept update/drift: This is the ability to continually perform well at a given task T_i by utilizing additional training data for T_i if it arrives at a later time (concept updating), or when new data for the changing environment arrives (concept drift). This process should only use the data of T_i to update the model while not forgetting other learned tasks.

Non-confusion: Machine learning algorithms often find the minimal set of discriminating features necessary for classification. Thus, when more tasks emerge for learning in our LLL setting, earlier learned features may not be sufficient, leading to confusion between classes. For example, after learning to distinguish between images of "1" and "0" as T_1 and T_2, the learned model may identify straight stroke for class "1" and curved stroke for "0". But after learning "I" and "O" as T_{10} and T_{11} for example without using data for "0" and "1", the model may confuse between T_1 and T_{10} ("1" and "I") and similarly between T_2 and T_{11} ("0" and "O") when the model is tested on all tasks learned so far. In human lifelong learning, this type of confusion may happen too. For example, when meeting new people, if the first two people are visually distinct (such as very tall vs. very short) we can rely on this feature to tell them apart. However, if more people arrive and they are similar to the first two people, we may initially confuse them and must find finer details to reduce confusion, or to uniquely distinguish them. In the extreme case where we encounter identical twins, significant effort may be required to learn the necessary details (by re-using their facial image data).

Human-like learning: As a new type of evaluation criteria for LLL, we can consider how well an approach is predictive of human learning behaviour. If a LLL approach or framework is also able to provide explanatory power and match peculiarities of human learning (such as confusion or knowledge transfer in certain scenarios), it would have value in fields outside of machine learning.

2.3. Overview of Different LLL Approaches

The mechanisms used to perform LLL tend to fall into three categories and often only demonstrate subsets of LLL properties as previously discussed.

The first mechanism, replay, commonly works by storing previous task data and training on it alongside new task data (Rebuffi et al., 2017; Isele & Cosgun, 2018; Chaudhry et al., 2019; Wu et al., 2019). As a result of its data and computation inefficiency, we consider it generally not to be very a human-like learning mechanism.

The second mechanism is regularization. This mechanism works by restricting weight changes (making them less "flexible") via a loss function so that learning new tasks does not significantly affect previous task performance (Kirkpatrick et al., 2016; Zenke et al., 2017b; Chaudhry et al., 2018; Ritter et al., 2018; Li & Hoiem, 2017; Zhang et al., 2020). We use this mechanism as the basis for our unified framework. However, instead of simply controlling weight flexibility to *retain* previous task performance, we leverage it to also encourage forward transfer.

The third mechanism, dynamic architecture, commonly works by adding new weights for each task and only allowing those to be tuned (Rusu et al., 2016; Yoon et al., 2018; Xu & Zhu, 2018). This is often done without requiring previous task data and completely reduces forgetting while also allowing previous task knowledge to speed up learning of the new task. While this mechanism is necessary for LLL of an arbitrarily long sequence of tasks (any fixed-size network will eventually reach maximum capacity), it should be used sparingly to avoid unnecessary computational costs. Our particular approach, described in Section 4, can control the rate at which the representational capacity of the neural network is used up, so that longer task sequences may be accommodated.

3. A Unified Framework for LLL

In this section we describe how our unified framework works. We start by introducing the central mechanism and in the rest of the section, discuss how to use the central mechanism to support the multiple desirable LLL properties described in Section 2.2. For a more extensive treatment on achieving these properties with this unified framework, see (Ling & Bohn, 2019). In the pseudo-codes provided, the lines where the central mechanism is applied is marked by a ●. In Section 4 we discuss a particular realization of this framework to experiment on.

3.1. A Central Consolidation Mechanism

We propose a lifelong learning framework which situates a consolidation policy as the central mechanism. The consolidation policy works through a single dynamic hyperparameter, $\boldsymbol{b}$, which separately controls the flexibility of *all* network weights. In other words, each weight in the network can have its own consolidation value, representing how easy (or hard) it is to modify the weight. Depending on the specific $\boldsymbol{b}$-setting policy used during training, several desirable learning properties can be achieved. While the individual weights of the network are learned via back-propagation, $\boldsymbol{b}$ is set by a consolidation policy (which depends on the specific approach).

The central consolidation mechanism ultimately works through dynamically modifying the loss function. To describe more precisely, we will deviate a little from the no-

tation used in (Kirkpatrick et al., 2016), which introduced the Elastic Weight Consolidation LLL approach. If each network weight, θ_i, is associated with a consolidation value of $\boldsymbol{b}_i \geq 0$, the loss for the new task by itself, L_t, is combined with weight consolidation as follows:

$$L(\theta) = L_t(\theta) + \sum_i \boldsymbol{b}_i(\theta_i^t - \theta_i^{target})^2 \quad (1)$$

Here, θ_i^{target} is the target value for a weight to be changed to. This can be either its value before training of the new task, or zero, in the case where we explicitly want to prevent certain weights from being used. θ_i^t is the weight value being updated during training on task t. The loss now had the following behaviour: a large $\boldsymbol{b}_i$ value indicates that changing weight θ_i away from θ_i^{target} is strongly penalized during training. When a set of weights have large corresponding $\boldsymbol{b}$ values, we will often refer to them as **"frozen"**. In contrast, value of $\boldsymbol{b}_i = 0$ indicates that the weight is free to change. We refer to these weights as **"unfrozen"**. If $\boldsymbol{b}_i$ is arbitrarily large, we can consider θ_i to be masked during backpropagation and completely prevented from changing to improve efficiency.

3.2. Continual Learning of New Classification Tasks

In both lifelong and human learning, we desire to learn new tasks after learning previous tasks. In humans, this is enabled by the ability to continually grow new connections between neurons and remove old connections. When this ability is compromised, so is our ability to learn new things. Similarly, in our conceptual framework we consider learning new tasks with the help of network expansion and pruning.

The pseudo-code in Algorithm 1 describes how to learn a new task, T_k, in a deep neural network in our conceptual framework after previous tasks $T_1, ...T_{k-1}$ have been learned. The role of the consolidation policy in this algorithm (and all others in this paper), as marked by ●, is to ensure that newly added weights have the proper flexibility and that previous task weights will be inflexible, to prevent forgetting (Section 3.3).

This is a very general, high-level framework, and in subsequent sections, several of the steps in Algorithm 1 will be explained and expanded upon.

3.3. Non-Forgetting

Maintaining performance on previous classification tasks while learning new concepts is the primary difficulty lifelong learning approaches aim to combat. In our framework, we can design consolidation policies to make sure previous tasks will not be forgotten while the new task is learned with the data for the new task only. An intuitive way to prevent forgetting is by using a larger $\boldsymbol{b}$ value for weights which most influence the loss of a trained model (Kirkpatrick et al., 2016).

Algorithm 1 Continual Learning of New Tasks (● indicates consolidation policy)

`// Given that tasks` $T_1, ..., T_{k-1}$ `have been learned`
Recruit free units for T_k `// this can be all available units, or determined by the task difficulty and similarity with previous tasks. See Section 3.4`
● Initialize weights from earlier units to newly recruited units `// see green links in Fig. 2 and relating to forward transfer, see Section 3.4`
● Initialize weights for the new units `// see blue links in Fig. 2`
● Set consolidation values for non-forgetting of previous tasks `// see red links in Fig. 2. For non-forgetting, see Section 3.3`
Train the new task T_k to minimize Eq. 1 `// only on the data of new task` T_k
● Optionally prune the network for T_k `// prune to free units for future tasks and graceful forgetting).`

Our framework is inspired by the influential Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2016), which essentially uses the second derivative of the loss with respect to each weight as the weight consolidation value. Intuitively, well-learned previous classification tasks usually have a sharp local minimum (compared with a randomly-initialized network for the new task), thus, have relatively large second derivatives and $\boldsymbol{b}$ values. However, in our framework, $\boldsymbol{b}$ can be set to any large values based on the policy (to be designed), which controls how much non-forgetting should occur, based on desired applications. If $\boldsymbol{b}$ is set to infinitely large, then forgetting would not happen at all, and this is equivalent to training a separate new network for the new classification task.

Pseudo-code reflecting how a consolidation policy can simply be used in our framework is given in Algorithm 2. The role of the consolidation policy here, similar to in Algorithm 1, is to prevent weights for previous tasks from being changed, but allowing the new task weights to change.

3.4. Forward Transfer

While Section 3.3 ensures previous tasks will not be forgotten during learning the new task, the previous tasks do not "help" learning the new task, a concept prominent in

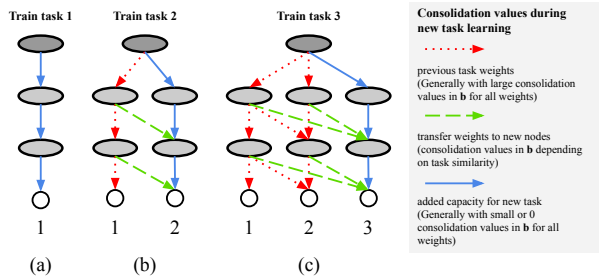

Figure 2. Network expansion and consolidation in our LLL framework. In step (a), task 1 is being trained. In step (b) task 2 is being learned without forgetting of task 1. In (c), task 3 is learned without forgetting of the previous tasks. Note that links represent the weights between groups of nodes and missing links indicate disconnected nodes (weights frozen with value of zero).

Algorithm 2 Non-forgetting

`// We need to learn a new task` T_k `by using data for only` T_k`, without forgetting` $T_1 ... T_{k-1}$

- Unfreeze only units for T_k `// blue and green links in Fig. 2 with` $\boldsymbol{b}$ `set to small values`
- Freeze all weights for tasks $\{T_1 ... T_{k-1}\}$ `// red links in Fig. 2, with` $\boldsymbol{b}$ `set to large values`

Train the new task T_k to minimize Eq. 1

multi-task and transfer learning (Pan & Yang, 2009; Zhang & Yang, 2017), and appears in LLL as "forward transfer". However, the question of how to utilize the previous task skills to accelerate new learning is a difficult and ongoing research question. In this conceptual framework discussion, we refer to the function estimating how much one task can be expected to help learn another as simply "similarity", $sim : \mathcal{T} \times \mathcal{T} \longrightarrow [-1, 1]$. This function is not necessarily commutative (i.e. $sim(T_i, T_j) \not\equiv sim(T_j, T_i)$[1]). If a new task is very similar (or identical in the extreme case) according to some metric such as visual similarity, ($sim \approx 1$) then the new task can be learned with no or little training data (few-shot learning). In other cases, previous tasks may actually impair the new task learning ($sim < 0$).

As summarized in Algorithm 3, positive forward transfer in our framework may be achieved through initialization of the weights for current task based on its similarity with previous tasks. For example, when initializing the output-layer weights for a new task, if the new task T_k is identical to a previous task T_j, then we can initialize the weights of T_k to reflect the values of those for T_j, requiring no or few new units for T_k. This idea is conceptually similar to that used by GO-MTL (Kumar & Daume III, 2012) and ELLA (Ruvolo & Eaton, 2013), where knowledge is selectively shared between related tasks. The role of the consolidation policy here, is to further allow positive transference and limit negative transference by controlling flexibility of weights between tasks (green links in Fig. 2).

[1]For example, consider the case where one task is actually a subset of another. Here, the more general task will be more helpful for the smaller task.

Algorithm 3 Forward Transfer

`// Assume` $T_1, ..., T_{k-1}$ `have been learned, and we need to transfer their knowledge while learning the new task` T_k

$\mathbf{s} = (sim(T_1, T_k), ..., sim(T_{k-1}, T_k))$

- Initialize weights for T_k based on $\mathbf{s}$ `// as weight initialization, see Fig. 3`
- Set values of $\boldsymbol{b}$ for T_k's links based on $\mathbf{s}$ Train the new task to minimize Eq. 1

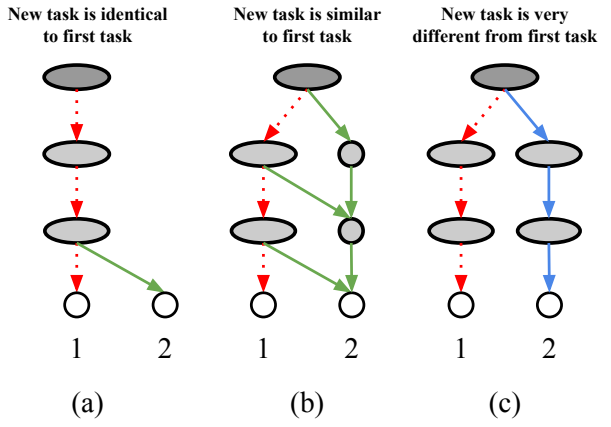

Figure 3. Weight initialization for different forward transfer cases. Green weights indicate the the weights are initialized to reflect those of the red weights with large values in $\boldsymbol{b}$. Blue weights are randomly initialized and have small values in $\boldsymbol{b}$. The width of the hidden layers reflect the relative number of nodes. In (a) is a special case where T_2 is identical to T_1 so no free units are needed, and weights can be "copied". In (b) is a case where T_2 is similar to T_1 in terms of training data or domain knowledge needed, so that a smaller number of free nodes are needed and weights of T_2 could be partially copied from T_1. In (c) is when negative transfer from T_1 to T_2 may happen or when tasks are unrelated.

4. LLL with Non-forgetting and Forward Transfer

In this section we will describe a particular LLL approach based on our unified framework which is designed to support non-forgetting and forward transfer with the single consolidation mechanism. While the framework described

in Section 3 is not concerned with how exactly to determine which weights of a neural network "belong" to a given task, that is the function we discuss first in Section 4.1. In Sections 4.2 and 4.3 we then describe consolidation strategies for non-forgetting and forward transfer respectively.

4.1. Identifying Critical Sub-networks (Lottery Tickets)

Central to our approach is the ability to identify a critical sub-network associated with each class learned by a neural network, where each class is associated with a single output dimension. We refer to these sub-networks as lottery tickets (LTs), taking inspiration from the lottery ticket hypothesis (Frankle & Carbin, 2018). The visual intuition behind our approach to identifying LTs is provided in Figure 4 – where the final LT weights for class 0 are shown with solid red edges in (e). Our particular approach is designed so that even when very sparse sub-networks are chosen, our method guarantees the presence of a full path (from input to output nodes) and it is not easy for non-LT weights to influence the output of the corresponding LT class.

To identify the weights making up a LT, we first obtain the rescaled magnitudes of all weights such that for each node, n, the maximum weight magnitude pointing to it is 1:

$$in(n) := \frac{|in(n)|}{max(|in(n)|)}, \tag{2}$$

where $in(n)$ is the set of weights pointing into a node. Next, we recursively identify the LT *nodes* in a network for class index c:

$$\begin{aligned} N_c(depth) &= \{c\} \\ N_c(l) &= \{n \in layer(l) | \\ &\quad (\exists w |\, (w \in out(n) \cap in(N(l+1)) \wedge w > \tau))\} \end{aligned} \tag{3}$$

where $N(l)$ is the set of important nodes for layer l, $out(\cdot)$ and $in(\cdot)$ are correspondingly the set of weights pointing out of and into of any nodes in the argument, and $\tau \in [0, 1]$ is the pruning threshold that defines how sparse the selected sub-networks are. The base case of $N_c(depth) = \{c\}$ is seen in Figure 4 as the highlighted output node for which the LT is identified. Given N, the set of LT weights for class c is finally defined by:

$$LT(c) = \{w \in layers | w \in in(N_c)\} \tag{4}$$

For the biases of a layer, the same condition is used. After the LT for a class has been identified, the set of weight locations responsible are stored in memory.

Given a LT, we define three specific types of weights: central LT weights, backward-incidental weights, and forward-incidental weights. Central LT weights are weights feeding into LT nodes which have normalized magnitudes $> \tau$, backward-incidental weights are those feeding into LT nodes which have normalized magnitudes $< \tau$, and forward-incidental weights are those leading out of LT nodes into non-LT nodes.

4.2. Non-forgetting

By design, LTs are critical for a neural networks performance on their respective classes and tasks. It follows that by preventing the weights of the LTs for a task from changing, the performance on the task can been maintained. We can implement this kind of weight-change penalty with $\mathbf{b}$ and Equation 1 as described in Section 3.1. The particular consolidation policy we use to achieve non-forgetting is:

$$\mathbf{b}_i = \begin{cases} b_{nf} & \text{if } \theta_i \in LT(C) \\ b_{fw} & \text{if } \theta_i \in out(N(C)) \\ 0 & otherwise \end{cases} \tag{5}$$

where C is the set of classes that have already undergone training. In other words, a consolidation value of $b_{nf} \geq 0$ is applied to central and backward-incidental LT weights, and $b_{fw} \geq 0$ is applied to forward-incidental weights.

Given a fixed pruning value, τ, modulating b_{nf} allows us to control how strongly past skills are maintained. Modulating b_{fw} however, has a larger effect on forward transfer, which we discuss next.

4.3. Forward Transfer

The ability to transfer knowledge from previous tasks to new ones, known as forward transfer, allows for more efficiently learning tasks later in a sequence. In our approach, forward transfer is supported by proper flexibility of forward-incidental weights (dashed green edges in Figure 4). In the consolidation policy described in Equation 5, this corresponds to small b_{fw} values. We also suspect that it will be important to ensure a sufficient amount of free weights for learning a new task by increasing the sparsity of LTs (corresponding to larger values for τ).

5. Experimental Setups and Results

For each of non-forgetting and forward transfer, we discuss the relevant metric, task, and then examine the experimental results. For both experiments presented, we evaluate on the Split-MNIST task sequence (Zenke et al., 2017a), which consists of five tasks: 0 vs 1, 2 vs 3, etc. Task orders are shuffled across trials for these experiments to remove any

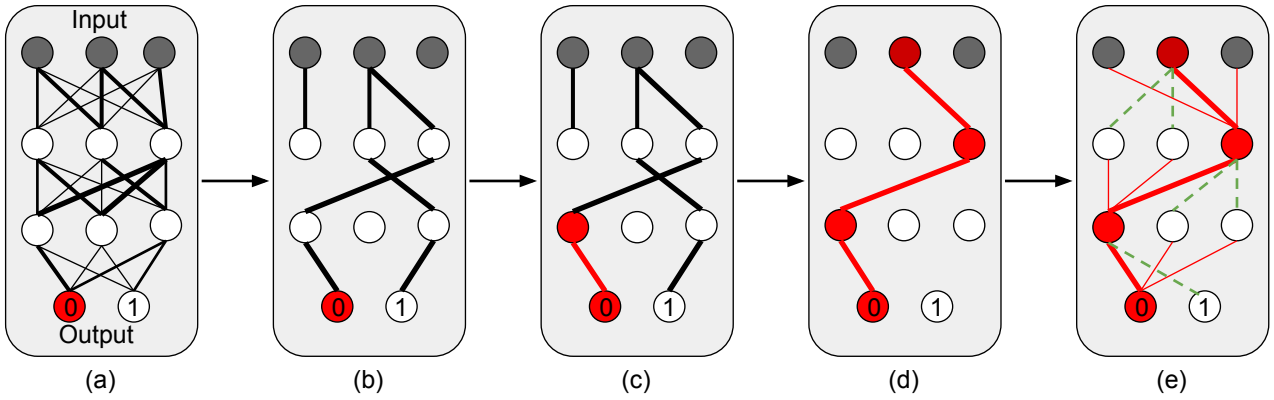

Figure 4. Lottery ticket sub-network identification. From (a) to (b): take absolute value of weight, normalize input weights per node, and filter weights by threshold (magnitude displayed with line thickness). Weights below threshold are not shown in (b). From (b) to (c): $N(depth - 1)$ consists of the red node in the penultimate layer. From (c) to (d): the remaining important LT nodes are identified by following the weights over the threshold. From (d) to (e), all LT weights are displayed in solid red edges. LT backward-incidental weights are shown with *thin* solid edges and LT forward-incidental weights are shown with green dashed edges. Central LT weights are shown with thick red edges. LT nodes are red.

confounding effects potentially caused by particular task orders. Additional hyperparameters and settings for the experiments are provided in Table 1

5.1. Non-forgetting

Metric. Catastrophic forgetting refers to the tendency for a neural network to forget old skills as new ones are presented. Given a sequence of tasks to solve, the extent of catastrophic forgetting (the opposite of non-forgetting) can be measured with *backward transfer interference* (BTI) (Riemer et al., 2018), which is the average difference between the *retained accuracy* (RA) of each task and the *learned accuracy* (LA). The LA for a task is the test accuracy as measured immediately after the task is learned, while RA is the test accuracy as measured after all tasks have been learned. Intuitively, catastrophic forgetting will cause the retained accuracy for old tasks to be much smaller than the learned accuracy. When $BTI < 0$, we consider forgetting (or negative backward transfer) to have occurred. When $BTI > 0$ (old task skills improve as new ones are presented) we consider positive backward transfer to have occurred. In this work, we will report a *normalized backward transfer interference* score, $nBTI$:

$$nBTI = \frac{RA - LA}{1 - LA} \tag{6}$$

The $nBTI$ allows us to put models on a more equal footing when comparing backward transfer/non-forgetting. Consider the following two models whose error rates suffer proportionally from forgetting: (1) where $LA = 0.99$ and $RA = 0.98$ ($BTI = -0.01$), and (2) where $LA = 0.9$ and $RA = 0.8$ ($BTI = -0.1$). For both models, $nBTI = -100\%$, indicating that the error rate doubles. In our non-forgetting experiment, $nBTI$ is calculated using multi-head evaluation, so that at test time, the task index is known to the model.

Experiment: pruning and LT consolidation. Figure 5 demonstrates how τ and b_{nf} can work together for non-forgetting. As we expect, there is a clear trend for non-forgetting to improve as consolidation of LT weights increases in strength. However, once τ is below 0.95, $nBTI$ does not appear to be sensitive to changes in the hyperparameter. For experimental setup, refer to non-forgetting in Table 1.

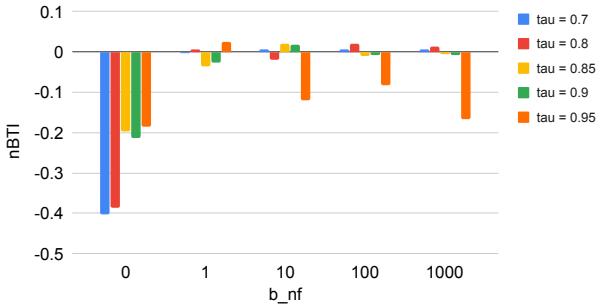

Figure 5. Non-forgetting experiment. The effect of τ and b_{nf} on $nBTI$.

5.2. Forward Transfer

Metric. Forward transfer refers to the ability for a model to learn new skills more easily by utilizing previously learning knowledge (and possibly with fewer weights needed) as

Table 1. Experimental setups for experiments in this section. For all experiments, the following hyperparameters are shared: batch size 64, 10 training epochs, 10-shot learning is used (unless otherwise specified), and training is done with the Adam optimizer (Kingma & Ba, 2014) with default Keras (Chollet et al., 2015) settings. Fully connected feed-forward neural networks with the specified hidden-layer widths and $relu$ activations are also used.

experiment	task	task shuffling	trials	tau	b_nf	b_fw	architecture
non-forgetting	Split-MNIST	TRUE	10	[0.95, 0.9, 0.85, 0.8, 0.7]	[0, 1, 10, 100, 1000]	0	[256, 256]
forward transfer	Split-MNIST	TRUE	10	[0.95, 0.9, 0.85, 0.8, 0.7]	[0, 1, 10, 100, 1000]	[0, 0.1, 1, 10]	[256, 256]

more tasks are learned. To measure forward transfer, we build off of FTI (forward transfer and interference) (Lopez-Paz & Ranzato, 2017), which is the difference between the LA for a task in the lifelong setting, and the LA for a task when learned independently ($LA_{\text{non LLL}}$). When $FTI < 0$, we consider negative forward transfer to occur, and when $FTI > 0$ we consider positive forward transfer to occur. However, to make the metric independent of the overall performance of the model, we report a *normalized forward transfer* metric:

$$nFTI = \frac{LA - LA_{\text{non LLL}}}{1 - LA_{\text{non LLL}}} \tag{7}$$

To provide some intuition behind this augmented forward transfer metric: when $nFTI = -1$: using the lifelong learning approach causes the learned error for each task to increase by 100% (2X worse), when $nFTI = 0$, using non-LLL is the same as LLL with regard to learned accuracy, and when $nFTI = 0.5$: learned errors rates are decreased by half.

Experiment: sparsity and consolidation. In Figure 6, we can observe the effect of τ and consolidation on $nFTI$. Most notably, when $b_{nf} = 0$, or τ is near 1, $nFTI$ is maximized. In contrast, this is when $nBTI$ is minimized. Additionally, $nFTI$ does prove to be sensitive to b_{fw}, especially when b_{nf} is large, suggesting that when forward-incidental weights are not free to adapt to the new task, modifying LT weights suffices.

		b_nf				
b_fw	tau	0	1	10	100	1000
0	0.7	-0.04	-0.28	-1.12	-2.63	-2.21
	0.8	-0.09	-0.26	-0.67	-1.84	-2.55
	0.85	-0.10	-0.22	-0.70	-1.46	-1.55
	0.9	-0.13	-0.23	-0.46	-0.83	-0.95
	0.95	-0.16	-0.08	-0.22	-0.20	-0.12
1	0.7	0.06	-0.41	-1.20	-2.15	-3.09
	0.8	0.00	-0.36	-1.25	-1.99	-2.10
	0.85	-0.09	-0.35	-0.87	-1.39	-2.15
	0.9	-0.15	-0.32	-0.74	-1.29	-1.41
	0.95	-0.09	-0.14	-0.50	-0.21	-0.27
10	0.7	-0.06	-0.32	-2.21	-3.30	-3.00
	0.8	-0.07	-0.34	-2.18	-2.92	-3.37
	0.85	-0.07	-0.33	-2.02	-2.13	-2.19
	0.9	-0.07	-0.46	-1.73	-2.50	-2.49
	0.95	-0.14	-0.42	-0.63	-1.04	-0.79

Figure 6. Forward transfer experiment. The effects of τ, b_{nf}, and b_{fw} on $nFTI$.

6. Conclusions

In this work, we presented a unified framework for lifelong learning that can be used to display may properties with a single central mechanism. Starting from this conceptual framework, we then describe a concrete approach designed to support non-forgetting and forward transfer. In particular, the approach performs weight consolidation guided by identified critical sub-networks – which we refer to as LTs. This approach tackles a difficult problem that captures multiple important aspects of human learning. Future work includes improving the sub-network identification algorithm to better support both non-forgetting and forward transfer at the same time, and implementing further consolidation-based skills as outlined in (Ling & Bohn, 2019). Progress in this area is critical for the development of computationally efficient and flexible machine learning algorithms.

References

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

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. S. Riemannian walk for incremental learning: Understanding forgetting and intransigence. *CoRR*, abs/1801.10112, 2018. URL http://dblp.uni-trier.de/db/journals/corr/corr1801.html#abs-1801-10112.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019.

Chollet, F. et al. Keras, 2015.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Fei-Fei, L., Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Isele, D. and Cosgun, A. Selective experience replay for lifelong learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Kamilaris, A. and Prenafeta-Boldú, F. X. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks, 2016. URL http://arxiv.org/abs/1612.00796. cite arxiv:1612.00796.

Kumar, A. and Daume III, H. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Ling, C. X. and Bohn, T. A conceptual framework for lifelong learning, 2019.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. A survey on deep learning in medical image analysis. *Medical image analysis*, 42: 60–88, 2017.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *NIPS*, pp. 6467–6476, 2017. URL http://dblp.uni-trier.de/db/conf/nips/nips2017.html#Lopez-PazR17.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. *The psychology of learning and motivation*, pp. 109–165, 1989.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359, 2009.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. icarl: Incremental classifier and representation learning. In *CVPR*, pp. 5533–5542. IEEE Computer Society, 2017. ISBN 978-1-5386-0457-1. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#RebuffiKSL17.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pp. 3738–3748, 2018.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL http://dblp.uni-trier.de/db/journals/corr/corr1606.html#RusuRDSKKPH16.

Ruvolo, P. and Eaton, E. Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pp. 507–515, 2013.

Shen, D., Wu, G., and Suk, H.-I. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Thrun., S. A lifelong learning perspective for mobile robot control. *Intelligent Robots and Systems*, 1995.

Thrun, S. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. Large scale incremental learning, 2019.

Xu, J. and Zhu, Z. Reinforced continual learning. In *Advances in Neural Information Processing Systems*, pp. 899–908, 2018.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. In *ICLR (Poster)*. OpenReview.net, 2018. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2018.html#YoonYLH18.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In Precup, D. and Teh, Y. W. (eds.), *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017a. URL http://dblp.uni-trier.de/db/conf/icml/icml2017.html#ZenkePG17.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In Precup, D. and Teh, Y. W. (eds.), *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017b. URL http://dblp.uni-trier.de/db/conf/icml/icml2017.html#ZenkePG17.

Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., and Kuo, C.-C. J. Class-incremental learning via deep model consolidation. In *The IEEE Winter Conference on Applications of Computer Vision*, pp. 1131–1140, 2020.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.