Split-and-Bridge: Adaptable Class Incremental Learning within a Single Neural Network

Jong-Yeong Kim, Dong-Wan Choi
Department of Computer Science and Engineering, Inha University, South Korea
kjy93217@naver.com, dchoi@inha.ac.kr

Abstract

Continual learning has been a major problem in the deep learning community, where the main challenge is how to effectively learn a series of newly arriving tasks without forgetting the knowledge of previous tasks. Initiated by *Learning without Forgetting* (LwF), many of the existing works report that knowledge distillation is effective to preserve the previous knowledge, and hence they commonly use a soft label for the old task, namely a *knowledge distillation (KD) loss*, together with a class label for the new task, namely a *cross entropy (CE) loss*, to form a composite loss for a single neural network. However, this approach suffers from learning the knowledge by a CE loss as a KD loss often more strongly influences the objective function when they are in a competitive situation within a single network. This could be a critical problem particularly in a *class incremental* scenario, where the knowledge across tasks as well as within the new task, both of which can only be acquired by a CE loss, is essentially learned due to the existence of a unified classifier. In this paper, we propose a novel continual learning method, called *Split-and-Bridge*, which can successfully address the above problem by partially splitting a neural network into two partitions for training the new task separated from the old task and re-connecting them for learning the knowledge across tasks. In our thorough experimental analysis, our Split-and-Bridge method outperforms the state-of-the-art competitors in KD-based continual learning.

Introduction

In recent years, deep neural networks (DNNs) have performed remarkably well in many practical applications like image or voice recognition (He et al. 2016),(Graves, Mohamed, and Hinton 2013), object detection (He et al. 2020), and language translation (Sutskever, Vinyals, and Le 2014). A typical DNN learns the entire data for a fixed number of target tasks at once. However, in real-life applications encountering a dynamic stream of samples such as autonomous robots and unmanned vehicles, it is necessary to continuously incorporate a series of new tasks into the model being trained.

This problem is known as continual learning (Parisi et al. 2019), which aims to incrementally train a model so that it can perform well on both previous tasks and new tasks. A major difficulty of continual learning in DNNs lies in the fact that the knowledge previously learned for old classes can severely be lost during the training process of new classes, often referred to as *catastrophic forgetting* (McCloskey and Cohen 1989; McClelland, McNaughton, and O'Reilly 1995). In addition, such a loss of previous knowledge can be even more aggravated in *class incremental learning* (CIL) where a single classifier should be incrementally unified. This is unlike *task incremental learning* (TIL) where an inference is usually made with a *task identity*, i.e., the prior knowledge of which task each sample belongs to, for as many classifiers as tasks that have been trained.

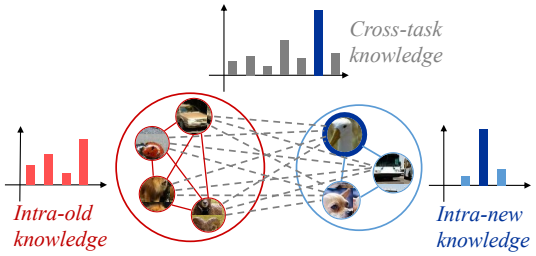
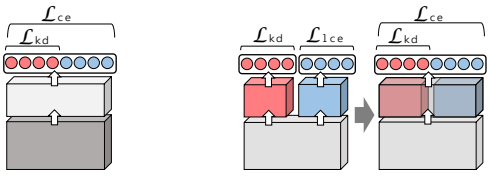


Figure 1: Three types of knowledge in CIL

In general, we can think of CIL as the problem of learning three types of knowledge, namely *intra-old*, *intra-new*, and *cross-task*, as shown in Figure 1. Either of the intra-old and the intra-new knowledge indicates how to discriminate classes only within the old task or the new task. On the other hand, the cross-task knowledge is about distinguishing every class in a particular task from the ones in the other task. Thus, we need all of the intra-old, intra-new, and cross-task knowledge for a unified classifier in CIL, but the cross-task knowledge is not necessary in TIL with a task identity provided at inference time. In this context, the goal of CIL is (i) to newly learn the intra-new knowledge as well as (ii) to incrementally update the cross-task knowledge (iii) without forgetting the previous intra-old knowledge.

As firstly introduced by *Learning without Forgetting* (LwF) (Li and Hoiem 2016, 2018), it has been reported that *knowledge distillation* (KD) (Hinton, Vinyals, and Dean 2015) is an effective strategy to preserve the intra-old knowledge. Also, the *rehearsal* technique (Lopez-Paz and Ranzato 2017; Rebuffi et al. 2017) is often supplementary used to acquire the cross-task knowledge as well as to give an additional opportunity for learning the intra-old knowledge. The intra-new



(a) Standard KD-based CIL (b) Split-and-Bridge

Figure 2: Standard KD-based CIL vs. Split-and-Bridge

knowledge can obviously be learned by a given task-specific dataset. Many of the KD-based continual learning methods follow this standard training scheme shown in Figure 2(a) even though they propose their own strategies to improve the way of making inference on the model already trained (Rebuffi et al. 2017; Wu et al. 2019; Zhao et al. 2020). More specifically, for the intra-old knowledge, they mainly use a *KD loss* with the soft label from the previous model along with a *cross entropy (CE) loss* with the class label from stored exemplar samples, and learn the cross-task knowledge and the intra-new knowledge from a CE loss with the class labels from exemplar and new samples.

Unfortunately, to be shown in our experiments, training a neural network is often more strongly influenced by a KD loss than by a CE loss. This is intuitively because the KD loss is not intended to newly learn additional information but for the network to stay as it is. Also, in terms of the quantity of information, a soft label used by the KD loss carries a larger amount of information than a one-hot encoded class label in the CE loss. Consequently, the aforementioned KD-based method suffers from learning the intra-new and cross-task knowledge, both of which can only be acquired by a CE loss, due to the interference from a KD loss for the intra-old knowledge.

One natural solution for this problem would be to train only samples of new classes separately in another neural network, and then combine it into the model previously trained for old classes. To this end, we need to minimize two KD losses, each of which transfers the knowledge from either model being integrated. This way seems to effectively learn the intra-new knowledge without any intervention. At the same time, however, two KD losses for the integration would doubly disrupt learning the cross-task knowledge from a CE loss, not to mention that we need such an extra model only temporarily used for new classes.

In this paper, we propose a novel CIL method, called *Split-and-Bridge*, which can effectively learn all three types of knowledge in a way that learning each type of knowledge is much less interfered by learning the others even within a single neural network. As presented in Figure 2(b), Split-and-Bridge works in the following two phases, that is, split and bridge. In the split phase, we partially split a previously trained network into two partitions exclusive in upper layers yet sharing the same component in lower layers. In the mean time, for each exclusive partition, we separately learn either the intra-new knowledge or the intra-old knowledge without interrupting each other. Then, in the bridge phase, we re-connect (i.e., *bridge*) those two partitions so that the cross-task knowledge can also be effectively learned without using two KD losses to integrate two pre-trained networks, i.e., the old model and new model.

In our experimental results, we show that Split-and-Bridge is superior to the state-of-the-art methods based on KD in CIL. In particular, our method turns out to be more adaptable to new classes than the existing methods, which is observed by the fact that the accuracy on the intra-new knowledge is fairly improved without loss of the knowledge previously learned.

Related Work

Continual learning in DNNs. Continual learning with DNNs has been mainly studied in the following two problem settings: *class incremental learning* (CIL) and *task incremental learning* (TIL). In CIL, a task identity should also be inferred for a unified classifier, whereas it is provided at test time in TIL. Both in CIL and TIL with DNNs, most of the existing works focus on how to overcome catastrophic forgetting, that is, the problem of preserving the previous knowledge when learning the new knowledge. The followings are the major branches of works on this problem strongly related to ours.

Rehearsal methods. *Rehearsal* methods (Lopez-Paz and Ranzato 2017; Rebuffi et al. 2017) store a subset of previous samples, and train them together with samples for a new task. This approach is known to be most effective to mitigate catastrophic forgetting in the CIL problem in which a single neural network needs to learn a sequence of tasks (van de Ven and Tolias 2019), and therefore many state-of-the-art methods complementary leverage a rehearsal method and so does our Split-and-Bridge method.

Parameter regularization. An alternative approach (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017; Chaudhry et al. 2018; Aljundi et al. 2018) to address catastrophic forgetting is to regularize the parameters of a neural network so that more important parameters with respect to the previous task can be protected during training on each new task. The main drawback of this approach lies in the fact that it suffers from learning a long sequence of tasks as its main strategy is not to further change parameters from themselves that have been already trained. Consequently, it is reported that these methods do not work well especially in CIL, compared to the methods based on knowledge distillation (KD) (van de Ven and Tolias 2019).

Parameter isolation. Mostly in a TIL scenario, we can also prevent catastrophic forgetting by separating model parameters for each task from the others. This idea motivates parameter isolation methods like *PNN* (Rusu et al. 2016), *DEN* (Yoon et al. 2018), *PackNet* (Mallya and Lazebnik 2018), *Piggyback* (Mallya, Davis, and Lazebnik 2018), *HAT* (Serrà et al. 2018), *CGATE* (Abati et al. 2020), all of which are designed to learn each task in an isolated part of the network and to make inference by using only the task-specific parameters selected by a task identity given at test time or inferred by an extra task classifier (Abati et al. 2020). Another type of parameter isolation is to temporarily learn a new task in

an extra network and then combine it into the previously learned model, which covers *P&C* (Schwarz et al. 2018), *DR* (Hou et al. 2018), *DMC* (Zhang et al. 2020), and *GD* (Lee et al. 2019). Our method is somewhat inspired by parameter isolation in the sense that we also temporarily split a network and then perform a unification process. However, none of the works do not deal with how to solve the CIL problem within a single neural network, which is the main goal of this work.

KD-based methods. Similar to parameter regularization, KD-based methods basically aim to retain a pretrained model by transferring the previous knowledge distilled from the model. However, they differ from parameter regularization in that KD allows parameters to be well updated during training on a new task to find a better optima for both the previous and new task. Since this approach was firstly introduced by *LwF* (Li and Hoiem 2016, 2018) for the TIL problem, there have been many variants following this standard LwF training scheme. *iCaRL* (Rebuffi et al. 2017) first proposes a combination approach of rehearsal and KD. More recent approaches tend to focus on data imbalance problem between old classes and new classes in CIL, which includes *WA* (Zhao et al. 2020), *Bic* (Wu et al. 2019), *LUCIR* (Hou et al. 2019), and *EEIL* (Castro et al. 2018). Although all these KD-based methods work effectively well to transfer the previous knowledge, we claim that these approaches could have undervalued the importance of the intra-new and cross-task knowledge, which should also be highly valued in the CIL problem we focus on.

Preliminary

This section first formally defines the class incremental learning (CIL) problem, and then describes the standard KD-based training method in the context of three essential types of knowledge in CIL as mentioned in the introduction.

Class incremental learning problem. We consider a sequence of tasks $T_1, T_2, ..., T_n$, where each T_i at the i-th time step is a set of classes such that $T_i \cap T_j = \emptyset$ for any $i \neq j$, and carries its task-specific sample set $\mathcal{D}_i$. Each $\mathcal{D}_i$ consists of pairs $(\mathbf{x}, \mathbf{y})$ of input sample $\mathbf{x}$ and its one-hot encoded label $\mathbf{y}$. At any t-th step, we are given the new sample set $\mathcal{D}_t$ as well as an exemplar set $\mathcal{M}_t$ of a fixed size, which is a subset of previously observed samples, i.e., $\mathcal{M}_t \subseteq \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_{t-1}$. Due to the limitation of memory space, it is usually assumed that $|\mathcal{M}_t| \ll |\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_{t-1}|$ in practice. Let Θ_i denote a neural network that have been trained from T_1 to T_i. Then, for each arrival of a new task T_t, the goal of the CIL problem is to newly train $\mathcal{D}_t$ with the help of $\mathcal{M}_t$ on the previously trained model Θ_{t-1} so that the resulting model Θ_t can work well with respect to all the classes in $T_1 \cup \cdots \cup T_t$.

As mentioned in the introduction, we can categorize the information required for each Θ_t into the three types of knowledge, namely *intra-new*, *intra-old*, and *cross-task*. The intra-new knowledge is essential to identify the class of each new sample in D_t from the other classes within T_t. Similarly, the intra-old knowledge represents how to discriminate samples within the set of old classes, i.e., $T_1 \cup \cdots \cup T_{t-1}$. On the other hand, the cross-task knowledge is necessary to distinguish samples across tasks, which informs how each new sample is different from old samples, and vice versa.

Loss	Accuracy (%)				
	Overall	**Old**	**New**	**Intra-old**	**Intra-new**
$\mathcal{L}_{ce}$	74.25	58.9	**89.6**	82.10	**90.85**
$\mathcal{L}_{kd}$ + $\mathcal{L}_{ce}$	**75.67**	**65.6**	85.75	**84.85**	88.30

Table 1: Four types of accuracy after incrementally learning two tasks, each consisting of 20 classes, on ResNet-18 with CIFAR-100, where the accuracy for the first task is 85.05%

Standard KD-based incremental learning. In the literature, KD-based CIL methods mostly adopt the same strategy to learn each of the above three knowledge types. To illustrate, we first denote $o(\mathbf{x})$ as the output logit from a neural network for a given input sample $\mathbf{x}$. Then, we can define a *cross entropy (CE) loss* on a model Θ for a given dataset $\mathcal{D}$ as:

$$\mathcal{L}_{ce}(\mathcal{D}, \Theta) = -\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbf{y} \log p(\mathbf{x}), \quad (1)$$

where $p(\mathbf{x})$ is the softmax probability vector of $o(\mathbf{x})$ such that $p(\mathbf{x})_i = \frac{e^{o(\mathbf{x})_i}}{\sum_j e^{o(\mathbf{x})_j}}$.

Similarly, we define a *knowledge distillation (KD) loss* as follows:

$$\mathcal{L}_{kd}(\mathcal{D}, \Theta) = -\sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \hat{q}(\mathbf{x}) \log q(\mathbf{x}), \quad (2)$$

where $q(\mathbf{x})$ is the softened probability of $o(\mathbf{x})$ such that $q(\mathbf{x})_i = \frac{e^{o(\mathbf{x})_i/\tau}}{\sum_j e^{o(\mathbf{x})_j/\tau}}$ for a temperature variable τ, and $\hat{q}(\mathbf{x})$ indicates the soft label of $\mathbf{x}$ from the referenced model of Θ, which is Θ_{t-1} at the t-th time step.

The standard KD-based method learns a new task T_t by minimizing the following *composite loss* function:

$$\lambda\, \mathcal{L}_{kd}(\mathcal{D}_t \cup \mathcal{M}_t, \Theta_t) + (1-\lambda)\, \mathcal{L}_{ce}(\mathcal{D}_t \cup \mathcal{M}_t, \Theta_t), \quad (3)$$

where λ is a hyperparameter balancing between two losses, and usually set to $\frac{C_{old}}{C_{old}+C_{new}}$ such that $C_{old} = |T_1 \cup \cdots \cup T_{t-1}|$ and $C_{new} = |T_t|$. We can further identify which part of this loss function is utilized to acquire each type of knowledge, and our understanding is as follows:

- Intra-old knowledge: $\mathcal{L}_{kd}(\mathcal{D}_t \cup \mathcal{M}_t, \Theta_t)$ + $\mathcal{L}_{ce}(\mathcal{M}_t, \Theta_t)$
- Intra-new knowledge: $\mathcal{L}_{ce}(\mathcal{D}_t, \Theta_t)$
- Cross-task knowledge: $\mathcal{L}_{ce}(\mathcal{D}_t \cup \mathcal{M}_t, \Theta_t)$

Thus, both the intra-new and the cross-task knowledge have to be learned by a CE loss, but this could not be very effective as a KD loss for the intra-old knowledge is likely to dominate the final loss function. This is a reasonable conjecture in that the soft label $\hat{q}(\mathbf{x})$ will carry a larger amount of information than the one-hot encoded label $\mathbf{y}$, leading to more updates by $\mathcal{L}_{kd}$ than by $\mathcal{L}_{ce}$.

To examine our conjecture, we conduct a simple experiment on learning CIFAR-100 (Krizhevsky, Hinton et al. 2009) on ResNet-18 (He et al. 2016) with two incremental tasks,

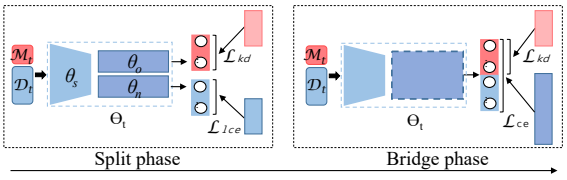


Figure 3: Two-phase learning of Split-and-Bridge

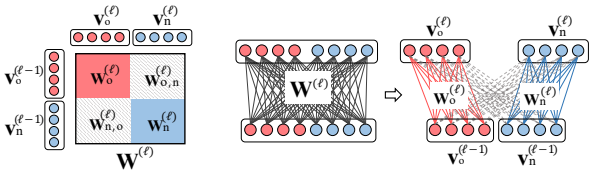


Figure 4: Sparsification across tasks

each consisting of 20 classes. Then, we compare the resulting accuracy of using $\mathcal{L}_{kd}$ + $\mathcal{L}_{ce}$ to that of using only $\mathcal{L}_{ce}$, where the accuracy is further divided into four categories; old, new, intra-old, and intra-new, to show how well each type of knowledge is learned by each loss function. When measuring either the intra-old or intra-new accuracy, we locally compare only the output probabilities corresponding to either the first (i.e., old) task or the second (i.e., new) task. As shown in Table 1, it is well observed that the new accuracy and the intra-new accuracy of using $\mathcal{L}_{kd}$ + $\mathcal{L}_{ce}$ gets lower than those of using only $\mathcal{L}_{ce}$. Thus, even though the standard KD-based method (i.e., $\mathcal{L}_{kd}$ + $\mathcal{L}_{ce}$) improves the overall accuracy partly because of the preservation of the intra-old knowledge, it is achieved at the sacrifice of the adaptability to the new task.

Proposed Adaptable Incremental Learning

In this section, we propose our *Split-and-Bridge* method for the CIL problem, which can effectively learn all three types of knowledge.

Motivation. As described in the preliminary section, the standard KD-based method suffers from learning the new knowledge by a CE loss due to the interference from the KD loss. With respect to the intra-new knowledge, a possible solution would be to separately learn a new task with an extra network, and then integrate its intra-new knowledge with the previous knowledge by distilling from two teacher models (i.e., the old model and the new model). However, at this time, the cross-task knowledge to be learned by another CE loss is doubly disturbed by these two KD losses.

This motivates us to propose a two phase learning method within a single network, where we first (i) *partially split* the network into two partitions for a separated learning, namely *split phase*, and then (ii) *re-connect* these trained partitions to additionally learn the cross-task knowledge, namely *bridge phase*. By doing so, we do not only learn the intra-task knowledge in an isolated partition without any competition between losses, but also acquire the cross-task knowledge without having to use double KD losses.

Split Phase

Separated learning within a single network. The goal of *split phase* is to learn the intra-new knowledge as independently as possible from the task of preserving the intra-old knowledge without using an extra neural network. To this end, we need to re-organize the given network Θ_t to have two disjoint branches at upper layers, denoted by θ_o and θ_n, coming from a shared common feature extractor spanning lower layers, denoted by θ_s, as shown in Figure 3. Once Θ_t is successfully transformed into the branched network, denoted by $\langle \theta_s, [\theta_o, \theta_n] \rangle_t$, we can learn both the intra-new and intra-old knowledge by the following loss function:

$$\mathcal{L}_{kd}(\mathcal{D}_t \cup \mathcal{M}_t, \langle \theta_s, \theta_o \rangle_t) + \mathcal{L}_{lce}(\mathcal{D}_t, \langle \theta_s, \theta_n \rangle_t). \quad (4)$$

Through $\mathcal{L}_{kd}$, we distill the intra-old knowledge from the previous model, i.e., Θ_{t-1}, into a part of the model consisting of the shared component followed by the disjoint partition for the old task, i.e., $\langle \theta_s, \theta_o \rangle_t$. Separately, the intra-new knowledge is learned on the other part of the model consisting of the shared component and the other partition, i.e., $\langle \theta_s, \theta_n \rangle_t$. Thus, with respect to θ_n and θ_o, each learning is completely independent, while both of learning processes are performed on θ_s. It is reasonable to have this shared component as the features at lower levels are usually applicable to every task and therefore such common features can effectively help to learn either task as well.

Note that we use a different type of loss, called *localized cross entropy (LCE)* and denoted by $\mathcal{L}_{lce}$, for learning the intra-new knowledge in this branched network. This loss is similar to the normal CE loss, but it only takes into account the local probabilities within the new task in order to focus on the intra-new knowledge as defined:

$$\mathcal{L}_{lce}(\mathcal{D}_t, \Theta) = - \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_t} \mathbf{y_t} \log p_t(\mathbf{x}), \quad (5)$$

where $p_t(\mathbf{x})$ represents a softmax output locally computed using only the sub-logit corresponding to the task T_t, and $\mathbf{y}_t$ is similarly a sliced hard label corresponding to T_t. Recall that $p(\mathbf{x})$ and $\mathbf{y}$ are defined for the all the classes in Eq. (1), but $p_t(\mathbf{x})$ and $\mathbf{y_t}$ are differently defined as $p_t(\mathbf{x})_i = \frac{e^{o(\mathbf{x})_i}}{\sum_j e^{o(\mathbf{x})_j}}$ *and* $\mathbf{y_t}_i = \mathbf{y}_i$ such that $i, j \in [C_{old} + 1, C_{old} + C_{new}]$, where C_{old} is the number of old classes and C_{new} is the number of new classes.

Weight sparsification across tasks. Then, how do we get a given network Θ_t to have a partially separated structure $\langle \theta_s, [\theta_o, \theta_n] \rangle_t$? In a simple way, we can randomly select two disjoint partitions and disconnect all the weights between them. However, such a simple method can cause a considerable loss of the previously trained knowledge. Our solution is instead to make those weights across partitions as sparse as possible during learning the intra-new knowledge as well as distilling the intra-old knowledge. This idea is inspired by *SplitNet* (Kim et al. 2017), which is originally introduced to build a tree-structured network with multiple branches such that similar classes are assigned to the same part of the network parameters. In the CIL problem setting, our goal is to prevent mutual interference between tasks during learning a new task, instead of grouping similar classes.

As illustrated in Figure 4, let us first consider a weight matrix $\mathbf{W}^{(\ell)}$ in layer ℓ, which takes the output $\mathbf{v}^{(\ell-1)}$ of the $(\ell-1)$-th layer as its input vector and produces its output $\mathbf{v}^{(\ell)}$ to be the input of the $(\ell+1)$-th layer. Also, let S denote the index of the last layer covered by θ_s, and then we have to split each $\mathbf{W}^{(\ell)}$ into two partitions for each layer $\ell \in [S+1, L]$, where L indicates the final layer of the network. Splitting each $\mathbf{W}^{(\ell)}$ means that we divide its input nodes $\mathbf{v}^{(\ell-1)}$ and output nodes $\mathbf{v}^{(\ell)}$ into two disjoint groups with a particular ratio (e.g., 1:1), namely $\mathbf{v}_o^{(\ell-1)}$ and $\mathbf{v}_o^{(\ell)}$ for the old task and $\mathbf{v}_n^{(\ell-1)}$ and $\mathbf{v}_n^{(\ell)}$ for the new task, and then disconnect all the weights between either $\mathbf{v}_o^{(\ell-1)}$ and $\mathbf{v}_n^{(\ell)}$ or $\mathbf{v}_n^{(\ell-1)}$ and $\mathbf{v}_o^{(\ell)}$ as shown in Figure 4. As mentioned above, to preserve the previous knowledge as much as possible in such a splitting process, we train a single network Θ_t by the following loss function, which can make weights to be disconnected as sparse as possible, while simultaneously trying to put each of the intra-old and the intra-new knowledge into two separated partitions:

$$\mathcal{L}_{kd}(\mathcal{D}_t \cup \mathcal{M}_t, \Theta_t) + \mathcal{L}_{lce}(\mathcal{D}_t, \Theta_t) + \gamma \sum_{\ell=S+1}^{L} (||\mathbf{W}_{o,n}^{(\ell)}||_2 + ||\mathbf{W}_{n,o}^{(\ell)}||_2), \quad (6)$$

where $\mathbf{W}_{o,n}^{(\ell)}$ and $\mathbf{W}_{n,o}^{(\ell)}$ indicate two sub-matrices whose weights should be disconnected, as defined:

$$\mathbf{W}_{o,n}^{(\ell)} = \{w_{ij} \in \mathbf{W}^{(\ell)} |\, i \in \mathbf{v}_o^{(\ell-1)} \wedge j \in \mathbf{v}_n^{(\ell)}\}$$
$$\mathbf{W}_{n,o}^{(\ell)} = \{w_{ij} \in \mathbf{W}^{(\ell)} |\, i \in \mathbf{v}_n^{(\ell-1)} \wedge j \in \mathbf{v}_o^{(\ell)}\}.$$

By Eq. (6), Θ_t is jointly optimized by $\mathcal{L}_{kd}$, $\mathcal{L}_{lce}$, and a regularization term for sparsification. Note that the regularization term imposes l^2 norm on two sub-matrices to be disconnected, and therefore minimizing this loss makes those weights as small as possible. γ controls the strength of this sparse regularization. Once the training of weight sparsification is performed enough, we explicitly disconnect all $\mathbf{W}^{(\ell)}$'s into two partitions $\mathbf{W}_o^{(\ell)}$'s and $\mathbf{W}_n^{(\ell)}$'s, all of which together constitute θ_o and θ_n, respectively, for further training on the resulting branched network by minimizing Eq. (4).

Note that this sparsification process can also help to prevent overfitting in earlier incremental steps. If we want our model to accept as many tasks as possible, it is somewhat unavoidable to start with a high-capacity model. However, such a large model tends to overfit by memorizing patterns of samples belonging to a few classes in earlier steps. Our regularization term for sparsification can mitigate this overfitting problem particularly when we train initial tasks consisting of only a few classes on a model with high capacity.

Adaptive split ratio. Now, the question is how much we allocate the partition of each task. In the final layer (i.e., when $\ell = L$), there is no choice but to assign the final outputs of old classes to $\mathbf{v}_o^{(L)}$ and set $\mathbf{v}_n^{(L)}$ to those of new classes. Other than the final layer, we propose an *adaptive splitting scheme* such that $|\mathbf{v}_o^{(\ell)}| : |\mathbf{v}_n^{(\ell)}| = \rho\, C_{old} : (1-\rho)\, C_{old} + C_{new}$, where ρ is a hyperparameter that controls the allocation rate for the old task, which depends on models, total number of steps, etc. Once $|\mathbf{v}_n^{(\ell)}| < 1$, we consider ℓ to be a layer shared by the intra-old and intra-new knowledge. In our experiments, we set ρ to a value between 1.0 and 1.4.

Algorithm 1: Split-and-Bridge Incremental Learning

1 $\Theta_0 \leftarrow$ a neural network randomly initialized;
2 $\mathcal{M}_1 \leftarrow \emptyset$;
3 **foreach** *incremental task* T_t **do**
  Input: model Θ_{t-1}, the task-specific data $\mathcal{D}_t$
  Output: model Θ_t
4 $\Theta_t \leftarrow \Theta_{t-1}$;
5 **if** $t = 1$ **then**
6 Train Θ_t by minimizing $\mathcal{L}_{ce}(\mathcal{D}_t, \Theta_t)$;
7 **else**
8 Train Θ_t by minimizing Eq. (6);
9 Explicitly disconnect Θ_t into $\langle \theta_s, [\theta_o, \theta_n] \rangle_t$;
10 Train $\langle \theta_s, [\theta_o, \theta_n] \rangle_t$ by minimizing Eq. (4);
11 Re-connect θ_o and θ_n to form Θ_t;
12 Train Θ_t by minimizing Eq. (3);
13 $\mathcal{M}_{t+1} \leftarrow$ random sample from $\mathcal{M}_t \cup \mathcal{D}_t$;

Bridge Phase

Given $\langle \theta_s, [\theta_o, \theta_n] \rangle_t$ that has separately learned the intra-old and intra-new knowledge, we re-connect two partitions θ_o and θ_n in order to learn the cross-task knowledge between them in the bridge phase. To this end, we first initialize all the weights that have been removed in the split phase to be zero. This is intuitively because a random initialization can yield erroneous cross-task information and this would happen to break the model $\langle \theta_s, [\theta_o, \theta_n] \rangle_t$ trained well in the split phase. Our intention in the bridge phase is to learn any new information across tasks on these zero-initialized *bridge* weights if necessary.

In order to train this re-connected network, we minimize the same loss function of Eq. (3) as the standard KD-based method. At this time, however, we train the model that has already learned the intra-new knowledge as well as the intra-old knowledge in the split phase, not a model trained only for the old task. Thus, since the intra-new knowledge is already there in the target model being trained, learning by Eq. (3) is like an auxiliary training process as for the intra-new knowledge.

In addition, the KD loss now transfers not only the intra-old knowledge but also the *common knowledge* between old and new classes as its referenced model is now $\langle \theta_s, \theta_o \rangle_t$ containing the shared component θ_s, not Θ_{t-1} owning only the intra-old knowledge. As a result, the KD loss would even help to learn the cross-task knowledge, which is mainly learned by the overall CE loss.

Overall algorithm. In summary, Algorithm 1 outlines how Split-and-Bridge trains a given neural network for each incremental task. Learning the first task is typical training by a CE loss (Lines 5-6). After that, we first train Θ_t to have a partially split network (Lines 8-9), and then perform a sep-

arated learning on it (Line 10). Finally, we re-connect the split partitions for further learning the cross-task knowledge (Lines 11-12). It is noteworthy that the KD loss in Lines 8 and 10 uses Θ_{t-1} as its reference model, but the reference model is changed to $\langle \theta_s, \theta_o \rangle_t$ for Line 12 once the branched network is successfully trained.

Experiments

Environment

Datasets and base model. In our experiments, we train two benchmark datasets, CIFAR-100 (Krizhevsky, Hinton et al. 2009) and Tiny-ImageNet (Le and Yang 2015), on ResNet-18 (He et al. 2016). CIFAR-100 consists of 50K training images, 500 per class, and 10K test images, 100 per class, from 100 classes in total, where every image is of size 32×32. We randomly arrange and divide the classes into a particular number of groups of the same size for each group to be an incremental task. Tiny-ImageNet includes 100K training images and 10K validation images for 200 classes, each of which is of size 64×64. As a labeled test set of Tiny-ImageNet is not available, we use its validation set as a test set. Similar to CIFAR-100, we randomly split the classes into a certain number groups of the same size to form a series of as many incremental tasks. As a base network, we use ResNet-18 for both datasets as it is widely used for various benchmark datasets including CIFAR-100 and Tiny-ImageNet in the literature (Devries and Taylor 2017; Keshari, Singh, and Vatsa 2019).

Compared methods. In order to evaluate the performance of Split-and-Bridge, we test the following KD-based class incremental learning methods: *iCaRL* (Rebuffi et al. 2017), *WA* (Zhao et al. 2020), and *Bic* (Wu et al. 2019). All three methods basically adopt the standard KD-based training method described in the preliminary section, which we name *STD*[1]. Their difference lies in their way of making inference. iCaRL makes a prediction by adopting k-NN classification on the exemplar set. Both WA and Bic, which are the state-of-the-art methods in CIL, propose new *balancing* techniques to overcome the data imbalance problem between old classes and new classes, yet commonly follow STD for their training scheme.

In addition to STD, we consider another baseline training method, called *double distillation (DD)*, which is the way of combining two neural networks separately trained for either the old task and the new task as described in the motivation of our proposed method. Note that any inference techniques are orthogonal to the training scheme itself this work focuses on, and therefore we apply WA to DD as well as Split-and-Bridge. Finally, we measure the performance of *oracle* as an upper bound, which is jointly trained at once using the entire dataset.

Implementation details. We implement all the methods[2] in PyTorch, and train each model on a machine with an NVIDIA TITAN RTX and Intel Core Xeon Gold 5122. In the split phase, we divide two last residual blocks along with the final fully-connected (FC) layer of ResNet-18 into two disjoint partitions, i.e., θ_o and θ_n, implying $S = 13$ and $L = 18$. Full details are covered in our supplementary material.

[1]The original version of iCaRL is slightly different from STD, but this STD version is often reported to perform even better than the original one by the recent works such as (Ahn and Moon 2020).

[2]https://github.com/bigdata-inha/Split-and-Bridge

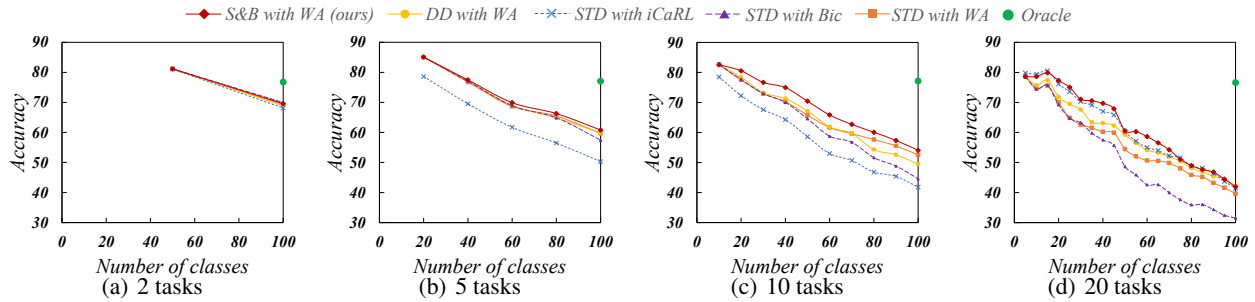
Number of tasks	2	5	10	20
CIFAR-100				
STD with iCaRL	68.14	59.50	55.6	60.04
STD with Bic	**69.96**	67.07	60.65	49.89
STD with WA	69.28	67.64	63.72	55.29
DD with WA	68.84	67.68	63.12	58.08
S&B with WA (ours)	69.6	**68.62**	**66.97**	**61.12**
Oracle	77.03			
Tiny-ImageNet				
STD with iCaRL	55.72	51.32	48.65	46.56
STD with Bic	58.16	55.23	48.47	43.81
STD with WA	57.96	55.97	51.61	47.57
DD with WA	58.33	56.80	53.12	48.14
S&B with WA (ours)	**60.52**	**57.16**	**54.81**	**51.63**
Oracle	62.35			

Table 2: Average accuracies over all the incremental tasks of ResNet-18 using CIFAR-100 and Tiny-ImageNet

Experimental Results

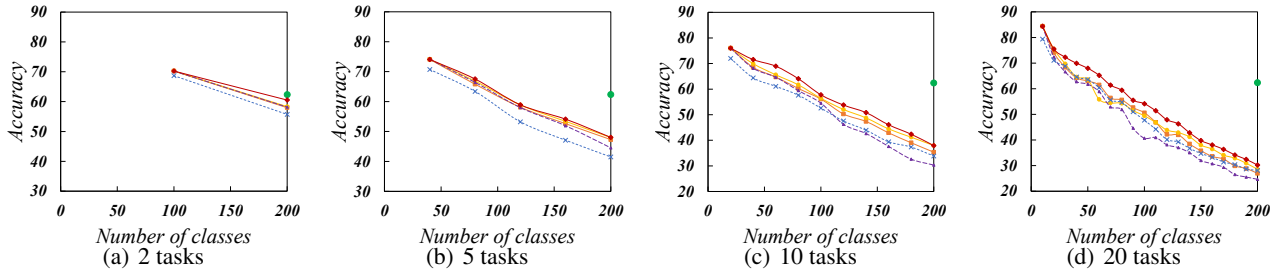
Overall performance. Table 2 summarizes the overall performance of all the compared methods, where we present the average accuracy of each method over all the incremental steps other than the first. It is clearly observed that our Split-and-Bridge method outperforms the other learning methods in most of the results. As also shown in Figures 5 and 6, Split-and-Bridge is not only the best on the average accuracy, but also consistently achieves the highest accuracy throughout the incremental steps in both CIFAR-100 and Tiny-ImageNet. The DD method occasionally performs better than STD probably because of learning the intra-new knowledge better, but its effectiveness turns out to be marginal in the sense that STD with WA or iCaRL often shows higher accuracies. This confirms our claim that a simple separated learning method with two neural networks does not work well in CIL.

Among the STD based methods, WA seems the best inference technique as it almost always shows the highest accuracy except for DD and Split-and-Bridge, and Bic is generally better than iCaRL in many cases. Interestingly, iCaRL performs quite well in the case of learning 20 tasks (i.e., 5 classes per task) in CIFAR-100, but both Bic and WA seem to struggle with learning this series of 20 tiny tasks as their performance gets abruptly worse than they used to be in the other cases. As observed in Figure 5(d), this is due to the fact that the accuracy of the methods except for iCaRL and Split-and-Bridge is far lower than expected in the first couple steps, and these inaccurately trained initial models badly influences learning the next sequence of tasks. As mentioned earlier, starting with a high-capacity model can inevitably cause an overfitting problem in earlier steps especially when each task consists of a few classes such as one of 20 tasks in CIFAR-100. Since iCaRL does not involve any FC layer to make an



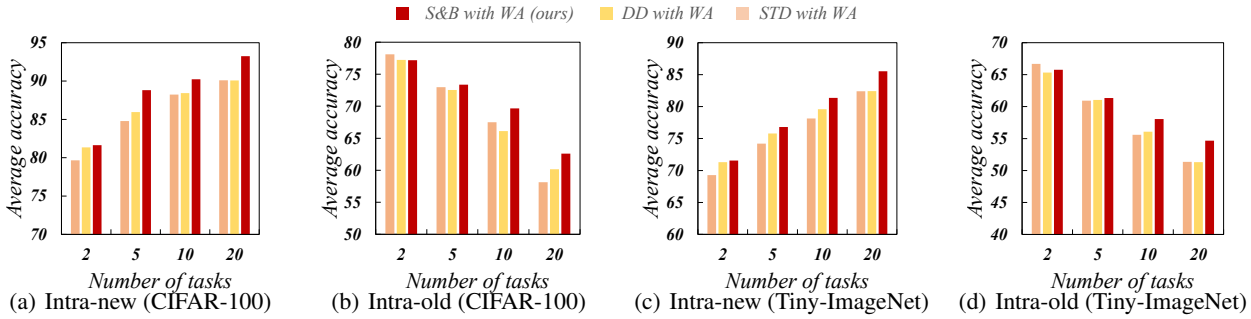
(a) 2 tasks (b) 5 tasks (c) 10 tasks (d) 20 tasks

Figure 5: Comparison on the accuracy for each incremental task using CIFAR-100



(a) 2 tasks (b) 5 tasks (c) 10 tasks (d) 20 tasks

Figure 6: Comparison on the accuracy for each incremental task using Tiny-ImageNet



(a) Intra-new (CIFAR-100) (b) Intra-old (CIFAR-100) (c) Intra-new (Tiny-ImageNet) (d) Intra-old (Tiny-ImageNet)

Figure 7: Comparison on the average intra-new and intra-old accuracy over the incremental tasks

inference, such an overfitting problem can be alleviated. Also, our Split-and-Bridge can mitigate overfitting with the help of regularization of the split phase, and hence still remains the best out of all the compared methods. For all the other details, please refer to our supplementary material.

Accuracy on the intra-new and the intra-old knowledge. As shown in Figure 7, we also compare the intra-new accuracy and the intra old accuracy of each method on the average. To focus on the effectiveness of each training scheme regardless of a balancing technique, we conduct this analysis using only the methods with WA. As expected, Split-and-Bridge always shows the best intra-new accuracy in both datasets, which tells our separated training scheme is quite effective to learn the intra-new knowledge. DD is also more adaptive to new tasks than STD when learning a small number of large tasks, but it gets less effective as the number of steps increases probably due to as many merging processes of two neural networks.

Another observation is that our Split-and-Bridge method is not only highly adaptive to new classes, but also good at preserving the intra-old knowledge as shown in Figures 7(b) and 7(d). This is contrast to the result of Table 1, where learning by a CE loss without any KD losses is also adaptive but prone to forgetting the previous knowledge. Through this analysis, it is confirmed that Split-and-Bridge is able to achieve a good placement between *stability* and *plasticity*.

Conclusion

In class incremental learning, we need to learn three types of essential knowledge, intra-new, intra-old, and cross-task, but the standard KD-based method has more focused on preserving the intra-old knowledge by sacrificing plasticity of a model. Motivated by this, we proposed a novel class incremental learning method, called Split-and-Bridge, which is highly adaptive to new classes yet stable enough not to forget too much of the previous knowledge. Through the extensive experiments on CIFAR-100 and Tiny-ImageNet, we confirmed that our Split-and-Bridge method can be an effective solution for the *stability-plasticity dilemma* in neural networks. As a future work, we hope to see our proposed training scheme is applied to a more complex deep learning problem such as object detection and sequence generation.

Acknowledgements

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2018R1D1A1B07049934) and in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (2019-0-00240, 2019-0-00064, and 2020-0-01389, Artificial Intelligence Convergence Research Center(Inha University)).

References

Abati, D.; Tomczak, J.; Blankevoort, T.; Calderara, S.; Cucchiara, R.; and Bejnordi, B. E. 2020. Conditional Channel Gated Networks for Task-Aware Continual Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 3930–3939. IEEE.

Ahn, H.; and Moon, T. 2020. A Simple Class Decision Balancing for Incremental Learning. *CoRR* abs/2003.13947. URL https://arxiv.org/abs/2003.13947.

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory Aware Synapses: Learning What (not) to Forget. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, volume 11207 of *Lecture Notes in Computer Science*, 144–161. Springer.

Castro, F. M.; Marín-Jiménez, M. J.; Guil, N.; Schmid, C.; and Alahari, K. 2018. End-to-End Incremental Learning. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, volume 11216 of *Lecture Notes in Computer Science*, 241–257. Springer.

Chaudhry, A.; Dokania, P. K.; Ajanthan, T.; and Torr, P. H. S. 2018. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, 556–572. Springer.

Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR* abs/1708.04552.

Graves, A.; Mohamed, A.; and Hinton, G. E. 2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, 6645–6649. IEEE.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2020. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.* 42(2): 386–397.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 770–778. IEEE Computer Society.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2018. Lifelong Learning via Progressive Distillation and Retrospection. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, volume 11207 of *Lecture Notes in Computer Science*, 452–467. Springer.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a Unified Classifier Incrementally via Rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 831–839. Computer Vision Foundation / IEEE.

Keshari, R.; Singh, R.; and Vatsa, M. 2019. Guided Dropout. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 4065–4072. AAAI Press.

Kim, J.; Park, Y.; Kim, G.; and Hwang, S. J. 2017. SplitNet: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 1866–1874. PMLR.

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. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114(13): 3521–3526.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Technical report, University of Toronto* 1–58.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N* 7.

Lee, K.; Lee, K.; Shin, J.; and Lee, H. 2019. Overcoming Catastrophic Forgetting With Unlabeled Data in the Wild. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 312–321. IEEE.

Li, Z.; and Hoiem, D. 2016. Learning Without Forgetting. In Leibe, B.; Matas, J.; Sebe, N.; and Welling, M., eds., *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, 614–629. Springer.

Li, Z.; and Hoiem, D. 2018. Learning without Forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* 40(12): 2935–2947.

Lopez-Paz, D.; and Ranzato, M. 2017. 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., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 6467–6476.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In Ferrari, V.; Hebert, M.; Sminchisescu, C.; and Weiss, Y., eds., *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, 72–88. Springer.

Mallya, A.; and Lazebnik, S. 2018. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 7765–7773. IEEE Computer Society.

McClelland, J. L.; McNaughton, B. L.; and O'Reilly, R. C. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* 102(3): 419.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, 109–165. Elsevier.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113: 54–71.

Rebuffi, S.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 5533–5542. IEEE Computer Society.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. *CoRR* abs/1606.04671.

Schwarz, J.; Czarnecki, W.; Luketina, J.; Grabska-Barwinska, A.; Teh, Y. W.; Pascanu, R.; and Hadsell, R. 2018. Progress & Compress: A scalable framework for continual learning. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 4535–4544. PMLR.

Serrà, J.; Suris, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming Catastrophic Forgetting with Hard Attention to the Task. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 4555–4564. PMLR.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 3104–3112.

van de Ven, G. M.; and Tolias, A. S. 2019. Three scenarios for continual learning. *CoRR* abs/1904.07734.

Wu, Y.; Chen, Y.; Wang, L.; Ye, Y.; Liu, Z.; Guo, Y.; and Fu, Y. 2019. Large Scale Incremental Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 374–382. Computer Vision Foundation / IEEE.

Yoon, J.; Yang, E.; Lee, J.; and Hwang, S. J. 2018. Lifelong Learning with Dynamically Expandable Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 3987–3995. PMLR.

Zhang, J.; Zhang, J.; Ghosh, S.; Li, D.; Tasci, S.; Heck, L. P.; Zhang, H.; and Kuo, C. J. 2020. Class-incremental Learning via Deep Model Consolidation. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, 1120–1129. IEEE.

Zhao, B.; Xiao, X.; Gan, G.; Zhang, B.; and Xia, S. 2020. Maintaining Discrimination and Fairness in Class Incremental Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 13205–13214. IEEE.