ScrollNet: Dynamic Weight Importance for Continual Learning

Fei Yang[1], Kai Wang[1], Joost van de Weijer[1]
[1] Computer Vision Center, Universitat Autònoma de Barcelona, Barcelona, Spain
{fyang,kwang,joost}@cvc.uab.es

Abstract

The principle underlying most existing continual learning (CL) methods is to prioritize stability by penalizing changes in parameters crucial to old tasks, while allowing for plasticity in other parameters. The importance of weights for each task can be determined either explicitly through learning a task-specific mask during training (e.g., parameter isolation-based approaches) or implicitly by introducing a regularization term (e.g., regularization-based approaches). However, all these methods assume that the importance of weights for each task is unknown prior to data exposure. In this paper, we propose ScrollNet as a scrolling neural network for continual learning. ScrollNet can be seen as a dynamic network that assigns the ranking of weight importance for each task before data exposure, thus achieving a more favorable stability-plasticity tradeoff during sequential task learning by reassigning this ranking for different tasks. Additionally, we demonstrate that ScrollNet can be combined with various CL methods, including regularization-based and replay-based approaches. Experimental results on CIFAR100 and TinyImagenet datasets show the effectiveness of our proposed method.

1. Introduction

In human life, knowledge is consistently acquired and accumulated. However, deep learning models often experience knowledge forgetting, a phenomenon known as catastrophic forgetting [23, 37], when they are exposed to a series of tasks. To address this issue, continual learning (CL) [41, 10, 34], also known as lifelong learning, has emerged as a vital research direction for a variety of learning and representation tasks (e.g., image classification[45, 34], semantic segmentation [12, 60], generative models [56], object detection, autonomous driving [50]). Continual learning aims to prevent the loss of previously acquired knowledge in neural networks over time.

This issue is intricately connected to the stability-plasticity dilemma [24, 39]. More precisely, when learning sequentially, the network needs to possess the ability to incorporate new knowledge (plasticity) while also maintaining stability to prevent forgetting previously learned tasks. Nevertheless, the stability-plasticity dilemma suggests that achieving both high plasticity and high stability simultaneously is challenging. Various continual learning approaches have been proposed to address this dilemma, which can be broadly categorized as follows: **regularization-based methods** [23, 2] add a regularization term to the objective function which impedes changes to the parameters deemed relevant to previous tasks; **replay-based methods** [42, 31] prevent forgetting by including data from previous tasks, stored either in an episodic memory or via a generative model; **parameter isolation-based methods** [45, 44] propose to modify the update rule of parameters of neural network to minimize the inter-task interference.

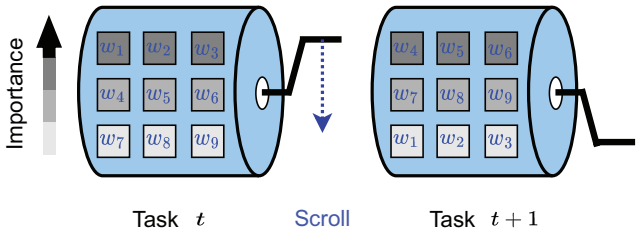

Figure 1. Our proposed method employs a dynamic network to pre-assign weight importance before training. This dynamic network ensures that the most crucial weights for a given task t are consistently assigned at the top of the 'roller' (a metaphor for a neural network). Once the current task is completed, we scroll the roller to readjust the order of weight importance for the subsequent task. This process aims to strike a balance between stability and plasticity, allowing the model to retain previously learned knowledge while remaining adaptable to new tasks.

The key to obtaining a balanced stability-plasticity tradeoff is to acquire the importance of weights for each task. This information can then be used to prioritize stability by penalizing changes in parameters crucial to previous tasks while allowing for plasticity in other parameters. The existing CL methods mentioned above determine the importance of weights for different tasks in two different ways: explicitly or implicitly. The explicit approach, such as parameter isolation-based methods [33, 32, 45], learns a

task-specific mask on parameters (or neurons) by introducing a sparsity loss. On the other hand, the implicit approach, like replay-based [43, 47, 30] and regularization-based [28, 23, 2] methods, utilizes memorized old data or regularization terms on objective functions to acquire the importance of weights. However, both approaches assume that the importance of weights for each task is unknown prior to data exposure.

In contrast to these approaches, we propose a novel method to pre-assign the weight importance for each task. Our inspiration comes from an easily overlooked characteristic of dynamic networks [16, 58, 20]. Dynamic networks were originally proposed for efficiency, enabling them to adapt their structures or parameters during inference. For example, they can allocate computations on demand by selectively activating model components (e.g., layers [54, 22, 4], channels [59, 21, 13], or sub-networks [29, 14, 25]) while avoiding performance degradation. The hidden characteristic under this property of dynamic networks is weight importance ranking. By splitting the network into small modules with different parameters, weight importance can be assigned by defining the composition of the sub-network in one dynamic network. For instance, the parameters involved in the smallest sub-network will be the most important ones for the current task. Note that sub-networks of different sizes can be mutually inclusive. This allows us to assign different importance to all parameters in the network even before touching training data. The final realization of this pre-assignment of weight importance is achieved through the optimization of the whole dynamic network.

Once we set the weight importance for one task, the next problem is how to use this information to achieve a balance between stability and plasticity in the continual learning of sequential tasks. Following the principle mentioned above, we need to re-order the weight importance before starting the next task. As shown in Figure 1, we propose a "scrolling" operation applied to the modules, which will assign the least importance to the parameters for the next task, which are the most important ones for the current task, and vice versa. Thus, we realize dynamic weight importance in sequential tasks. We name this novel and simple method *Scrollnet*. To the best of our knowledge, this is the first work that explores weight importance pre-assignment prior to data exposure in a continual learning setting. Additionally, we emphasize that *ScrollNet* is orthogonal to various CL approaches, such as regularization-based and replay-based methods.

In summary, the main contributions of this work are as follows:

- We propose *ScrollNet*, a novel continual learning method that can pre-assign the weight importance before starting a new task. Along with the proposed "Scrolling" strategy, ScrollNet can achieve dynamic weight importance and explicitly strike a balance between stability and plasticity.
- Our method is orthogonal to various continual learning approaches, and it improves performance when combined with them.
- We conduct extensive experiments on CIFAR100 and TinyImageNet. The experimental results demonstrate the effectiveness of our proposed method.

2. Related work

2.1. Continual learning

Continual learning [9, 34] methods can be loosely categorized into three groups of approaches: regularization-based, replay-based and parameter-isolation methods. We provide a brief overview of each approach below.

Regularization-based methods. The majority of these approaches add a regularization term to the loss function which impedes changes to the parameters deemed relevant to previous tasks. The difference depends on how to estimate relevance, and these methods can be further divided into data-focused [28] and prior-focused [23, 5, 2]. Data-focused methods use knowledge distillation from previously-learned models [30, 56, 28]. Prior-focused methods [23, 61] estimate the importance of model parameters as a prior for the new model. In recent times, there have been several notable studies that specifically concentrate on enforcing weight updates that lie within the null space of the feature covariance [52, 47]. In this paper, we apply our method over several regularization-based methods (including EWC [23], MAS [2] and LwF [28]) to verify our methods.

Replay-based methods. These approaches usually use memory and replay/rehearsal mechanism to recall a small episodic memory of previous tasks while training new tasks thus reduce the loss in the previous tasks. There are two main strategies: exemplar replay [5, 43, 47, 7, 6] and pseudo-replay [46, 56, 17, 30]. The former stores a few training samples (called exemplars) from previous tasks. The latter uses generative models learned from previous data distributions to synthesize data. In this paper, we also verify the effectiveness of our method combined with several exemplar replay-based methods, namely iCaRL [42], BiC [57], LUCIR [19], etc.

Parameter isolation-based methods. This branch tries to learn a sub-network for each task in a shared network [38]. In particular, Piggyback/PackNet [32, 33] iteratively assigns parameter subsets to consecutive tasks by constituting binary masks. SupSup [55] also finds masks in order to assign different subsets of the weights for different tasks. HAT [45] incorporates task-specific embeddings for

attention masking. Progressive Neural Network [44] allocates a sub-network for each task in advance and progressively concatenates previous sub-networks while freezing parameters allocated to previous tasks. [51] also proposes task-conditional hypernetworks for continual learning. [36] proposes nonoverlapping sets of units being active for each task. CCGN [1] proposes task-specific convolutional filter selection for continual learning, but they require a significantly large number of parameters to represent the masks for each task. This type of methods is also developed for the case where no forgetting is allowed in TFM [35]. In general, this branch is always restricted to the task-aware (task incremental) setting. Thus, they are more suitable for learning a long sequence of tasks when a task oracle is present.

Similar to parameter isolation-based approaches, our proposed method also segments the entire network into distinct sub-networks by employing a dynamic neural network architecture. However, in contrast to parameter isolation methods, our approach entails the direct and manual specification of "masks" for sub-networks, thereby explicitly assigning weight importance to each task even before touching the training data. Moreover, our method extends its applicability to the more general class-incremental learning setup. Importantly, our approach remains independent of regularization-based and replay-based approaches, establishing its orthogonality to those methods.

2.2. Dynamic networks

Different from the "dynamic networks (with sub-networks)" mentioned in Section. 2.1, which are specifically designed in various parameter isolation-based methods for overcoming catastrophic forgetting in continual learning, the related works we will briefly review here are about those that were originally proposed for inference efficiency. As opposed to static networks, dynamic networks can modify their structure or parameters and control the computation cost during inference in advance. Depending on the different types of "dynamic" in the architecture, the dynamic network methods can be grouped as dynamic depth, dynamic width, and dynamic routing.

Dynamic depth. The architecture with dynamic depth can reduce redundant computation by performing inference with variable depth. The realization approaches include *early exiting* [48, 4, 20, 22, 54], which involves executing only shallow layers for "easy" samples and avoiding the expensive computation cost of full layers, or layer skipping [15, 53, 49], which selectively skips intermediate layers depending on the complexity of input samples.

Dynamic width. By performing inference with variable *width* and comparing it with dynamic *depth*, the dynamic *width* architecture exhibits finer-grained control over computation costs. Although all layers are executed, multiple units (e.g., channels, neurons, or branches) within those layers can be selectively activated. Various implementations of the dynamic width architecture have been proposed, such as skipping neurons in fully-connected layers [3, 8], skipping branches in mixture-of-experts [21, 13], and skipping channels in CNNs [20, 59]. In this paper, we build our proposed *ScrollNet* upon [59].

Dynamic routing. The aforementioned approaches adapt the computation cost by adjusting the *depth* or *width* of the architectures. An alternative direction involves creating diverse network forms with multiple potential inference paths and conducting dynamic routing within these networks to adjust the computational graph for different samples. These related methods include path selection in multi-branch structures [40, 29] and tree-structured networks [25, 14].

3. Continual learning with a scrolling neural network

In this section, we describe our proposed dynamic network-based continual learning (CL) method, called *ScrollNet*. *ScrollNet* can directly assign the weight importance for each task with a dynamic neural network. After training for each task, the model will "scroll" the parameter importance assignments before training for the next task, which means reassigning the ranking of weight importance for different tasks. We emphasize that the proposed *ScrollNet* is orthogonal to regularization-based and replay-based CL approaches, which means it can be combined with those methods to achieve better performance.

Problem Statement. Consider a supervised continual learning scenario, a learner needs to solve T tasks sequentially without catastrophic forgetting of old tasks. We denote that $\mathcal{D}_t = \{\mathcal{X}_t, \mathcal{Y}_t\}$ is the dataset of task t, composed of a set of input images $\mathcal{X}_t$ and corresponding labels $\mathcal{Y}_t$. We assume a neural network $f(\cdot;\theta)$, parameterized by the model weights θ and a standard continual learning scenario aims to learn a sequence of tasks by minimizing the optimization problem at each step t:

$$\arg\min_\theta \mathcal{L}\left(f(\mathcal{X}_t;\theta), \mathcal{Y}_t\right), \tag{1}$$

where $\mathcal{L}(\cdot,\cdot)$ is a cross-entropy loss for image classification. $\mathcal{D}_t$ for task t is only accessible when learning task t. Note that replay-based continual learning approaches allow memorizing a small portion of data from old tasks.

To assign the weight importance for each task, we firstly split the weights θ into N non-overlapped sets: $\theta = \{\theta_{w_1}, \theta_{w_2}, ..., \theta_{w_N}\}$. We assume the list of these weight sets has a descending order of importance at the task t, then the optimization problem during training will be

$$\begin{aligned}\arg\min_\theta \mathcal{L}\left(f(\mathcal{X}_t;\{\theta_{w_1}, \theta_{w_2}, ..., \theta_{w_N}\}), \mathcal{Y}_t\right), \\ \text{subject to } I(\theta_{w_1}) > I(\theta_{w_2}) > .... > I(\theta_{w_N}),\end{aligned} \tag{2}$$

where $I(\cdot)$ stands for the importance score. In the next sections, we describe how to realize this assignment of weights ranking and how to reassign it in sequential tasks.

3.1. Weight importance assignment via a dynamic network

Dynamic networks aim to adapt their structures or parameters to the input during inference, therefore enjoy favorable properties which are absent in static models, such as efficiency, representation power, adaptiveness, etc. Rather than pursuing these characteristics of the dynamic network, here we leverage it to assign the ranking of weight importance. Specifically, we utilize a slimmable neural network [59], which comprises multiple sub-networks within $f(\cdot;\theta)$. To assign the desired weight ranking at the first task as shown in Eq. 2, we can predefine N sub-networks before training as

$$f_1(\cdot;\theta^1 = \theta_{w_1}), f_2(\cdot;\theta^2 = \theta_{w_1} \cup \theta_{w_2}), ..., f_N(\cdot;\theta^N = \theta). \tag{3}$$

Finally, we can realize this pre-assigned ranking by using the following loss during training:

$$\mathcal{L}_{dynamic} = \sum_{n=1}^{N} \mathcal{L}\left(f_n(\mathcal{X}_t;\theta^n), \mathcal{Y}_t\right). \tag{4}$$

As we can observe from this loss function, θ^1 serves as the foundational sub-network for task t and holds the highest significance. On the other hand, θ^2 and others, despite carrying more parameters, primarily serve to enhance the performance of this foundation, indicating a comparatively lesser importance. An illustration is presented in Figure 2 (see Task 1), where red connections represent one of the sub-networks with the most significant parameters, while the combination of red and green connections represents another sub-network with the first and second most important parameters.

3.2. Scrolling neural network for continual learning

In the previous section, we addressed the assignment and realization of weight importance ranking before and during training for each task. The challenge now lies in leveraging this knowledge to achieve a better stability-plasticity trade-off in continual learning. As mentioned earlier, the principle used for overcoming catastrophic forgetting is penalizing changes in parameters that are important for previous tasks while allowing updates to less important parameters. Building upon this principle, we propose a parameter "scrolling" approach before starting the current task. As shown in Figure 2, we always assign the most important parameters on the top/red connections (corresponding to the sub-network θ_1 in Eq. 4) and less important parameters on the bottom connections (i.e., weight importance decreases from red connections to blue connections). The operation of "scrolling" reassigns the less important parameters from the previous task to become the most important parameters for the current task, and vice versa for the most important parameters from the previous task. Consider the assignment of weight importance ranking at the first task, as shown in Eq. 3. The weights will be scrolled accordingly with scrolling step size S at task t as

$$\theta^1 = \theta_{w_{(t\%N)*S}}, \theta^2 = \theta_{w_{(t\%N)*S}} \cup \theta_{w_{(t\%N)*S+1}}, ..., \theta^N = \theta. \tag{5}$$

To implement this updated assignment, the same loss function as in Eq. 4 will be utilized during training for task t.

3.3. Combination of ScrollNet and continual learning methods

To demonstrate the orthogonality of our proposed *ScrollNet* to regularization-based and replay-based continual learning methods, we first rephrase Eq. 4 as

$$\mathcal{L}_{dynamic} = \mathcal{L}\left(f_N(\mathcal{X}_t;\theta^N), \mathcal{Y}_t\right) + \sum_{n=1}^{N-1} \mathcal{L}\left(f_n(\mathcal{X}_t;\theta^n), \mathcal{Y}_t\right). \tag{6}$$

Then we can see that the first term in the loss function of *ScrollNet* is a normal cross-entropy loss at each task which encompasses the entire network (equivalent to the loss function in Eq. 1). Additionally, it incorporates a series of cross-entropy losses applied to other sub-networks for realizing weight ranking. Here, we present two examples showcasing the combination of ScrollNet with LwF [28] and the combination of ScrollNet with EWC [23]:

(1) Loss function of **ScrollNet + LwF**:

$$\mathcal{L}_N\left(f_N(\mathcal{X}_t;\theta^N_{new}), \mathcal{Y}_t\right) + \lambda * \mathcal{L}_O\left(f_N(\mathcal{X}_t;\theta^N_{new}), f_N(\mathcal{X}_t;\theta^N_{old})\right) + \sum_{n=1}^{N-1} \mathcal{L}_N\left(f_n(\mathcal{X}_t;\theta^n_{new}), \mathcal{Y}_t\right), \tag{7}$$

where $\mathcal{L}_N$ and $\mathcal{L}_O$ represent the cross-entropy loss applied to the output of the new task and old task for new data, respectively. θ_{new} and θ_{old} denote the parameters of the current model and the frozen old model, respectively. λ is a hyperparameter that determines the weight of regularization strength. In our experiments, we set λ to 1.

(2) Loss function of **ScrollNet + EWC**:

$$\mathcal{L}_N\left(f_N(\mathcal{X}_t;\theta^N_{new}), \mathcal{Y}_t\right) + \sum_i \frac{\lambda}{2} * F^i(\theta^{N,i}_{new} - \theta^{N,i}_{old})^2 + \sum_{n=1}^{N-1} \mathcal{L}_N\left(f_n(\mathcal{X}_t;\theta^n_{new}), \mathcal{Y}_t\right), \tag{8}$$

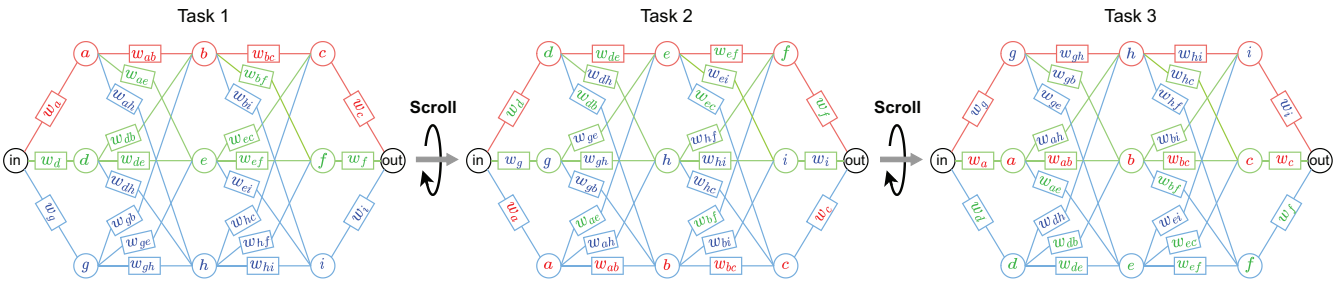

Figure 2. An illustration figure for our proposed *ScrollNet*. We use color-coded connections to represent the importance of parameters for each task. The red connections indicate the most important parameters, the green connections represent the second most important parameters, and the blue connections denote the third most important parameters. Before starting a new task, a "scroll" operation is performed, which reorders the parameter importance. This reassignment ensures that the upcoming task's most crucial parameters are appropriately positioned at the top of the network.

where F is the Fisher information matrix and i labels each parameter. We set λ to 5000 in our experiments.

Please see the pseudo algorithm. 1 for the summarized optimizing procedure of our *ScrollNet*.

Algorithm 1 Scrolling Neural Network (ScrollNet) for CL.

Input: Sequential data $\{\mathcal{D}_t\}_{t=1}^{\mathcal{T}}$, model weights θ
Input: Number of sub-networks N
Input: Loss function $\mathcal{L}_{CL}$ of the combined CL method
Input: Set scrolling step size S ▷ always 1 in this work
Randomly initialize θ
Equally split the model as $\theta = \{\theta_{w_1}, \theta_{w_2}, ..., \theta_{w_N}\}$
Initialize sub-networks as Eq. 3
for task $t = 1, ..., \mathcal{T}$ **do**
  Scroll the weights in each sub-network as in Eq. 5
  for batch $\mathbf{b}_t \sim \mathcal{D}_t$ **do**
    Set Total_loss = 0
    for $n = 1, ..., N$ **do**
      if $n == N$ **then**
        Calc $\mathcal{L} = \mathcal{L}\left(f_N(\mathcal{X}_t; \theta^N), \mathcal{Y}_t\right) + \mathcal{L}_{CL}$
        Total_loss += $\mathcal{L}$
      else
        Calc $\mathcal{L} = \mathcal{L}\left(f_n(\mathcal{X}_t; \theta^n), \mathcal{Y}_t\right)$
        Total_loss += $\mathcal{L}$
      end if
    end for
    Backpropagation with Total_loss
    Update model weights θ
  end for
end for

4. Experiments

4.1. Experimental settings

Datasets. We evaluate performance on two datasets: CIFAR100 [26], and TinyImageNet [27]. CIFAR100 contains 100 classes, each with 600 images, among which 500 images are for training and the other 100 are for test usage. Tiny ImageNet contains 100,000 images of 200 classes (500 for each class) downsized to 64×64 colored images. Each class has 500 training images, 50 validation images and 50 test images. In our experiments, we consider different numbers of dataset splits (e.g., 5 splits, 10 splits, and 20 splits) to verify the effectiveness of our proposed method on various lengths of sequential tasks.

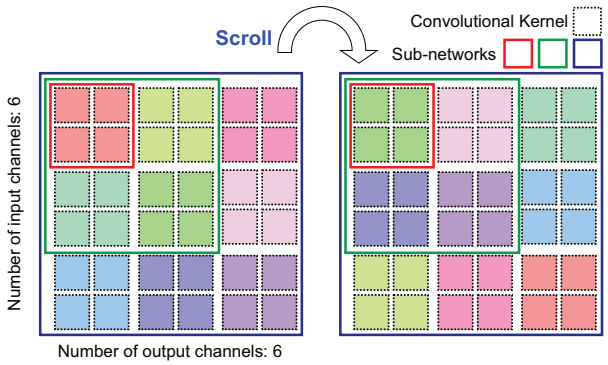

Figure 3. Illustration for one scrolling convolutional layer with three switchable widths as 2, 4, and 6.

Architecture. In our experiments, our proposed *ScrollNet* is built on ResNet18 [18], a commonly used architecture in the literature for CIFAR100 and TinyImageNet datasets. Firstly, following [59], we transform ResNet18 into a channel-wise dynamic network by incorporating slimmable convolutional layers, slimmable FC layers (used as classification heads), and switchable batch normalization layers instead of normal convolutional layers, FC layers, and standard batch normalization layers. In our experiments, we build the slimmable ResNet18 with two different numbers of sub-networks, 2 and 4. Subsequently, we implement *scrolling* for the slimmable convolutional layers and the slimmable FC heads, moving them in a predetermined direction when starting a new task, as depicted in Figure 3.

Please see Figure 2 for the *scrolling* FC layer. Note that we do not apply *scrolling* to switchable batch normalization layers since satisfactory performance was observed with a fixed order in sequential tasks. Finally, we implement two variants of our proposed method, namely ScrollNet-2 and ScrollNet-4, which involve 2 and 4 sub-networks (or can be viewed as the number of model splits) in the slimmable ResNet18, respectively. In this paper, we set the step size for scrolling constantly as $S = 1$.

Baselines. Besides the standard fine-tuning, to demonstrate the efficacy of the proposed *ScrollNet* mechanism, we integrate it with different continual learning approaches. Specifically, we combine *ScrollNet* with three exemplar-free methods such as EWC [23], MAS [2], and LwF [28]. Additionally, we also incorporate *ScrollNet* with three exemplar-based frameworks including iCaRL [42], LUCIR [19], and BiC [57]. We report average accuracy at each task on both *task-aware* and *task-agnostic* settings. Note that the implementation of all these baseline methods is from the CL framework[1] proposed in [34].

Training details. We train the model for 200 epochs per task for different numbers of splits. The learning rate is initialized to 0.1 and is decayed by a rate of 0.1 at the 80th and 120th epochs. We use an SGD optimizer with a batch size of 64. For the exemplar-based methods, we utilize 2000 exemplars selected with herding [42], constituting a fixed memory.

4.2. Main results

In Table 1 and Table 2, we present the average accuracy after the last task for all baseline methods, as well as their combinations with our proposed *ScrollNet*, on CIFAR100 and TinyImageNet, respectively. Additionally, Figure 4 and Figure 5 display the curves of average accuracies at each task on CIFAR100, representing the task-agnostic and task-aware settings, respectively.

From the tables, we observe that *ScrollNet*, when combined with various CL methods, outperforms most of their corresponding baselines in the task-agnostic setting across all different dataset splits, except for only LUCIR on TinyImageNet with 5 splits. Notably, *ScrollNet* demonstrates particularly promising results when paired with parameter regularization-based CL methods, such as EWC and MAS. For instance, when combined with EWC, the highest improvements reach up to 9.40% (5 splits), 8.47% (10 splits), and 5.61% (20 splits) on CIFAR100, as well as 2.67% (5 splits), 5.86% (10 splits), and 5.36%(20 splits) on TinyImageNet. It is worth mentioning that while EWC was not originally proposed for class incremental learning, the improvements observed in the task incremental learning setting (task-aware) are even higher, reaching 10.04% (5 splits), 12.40% (10 splits), and 16.64% (20 splits) on CIFAR100, and 6.70% (5 splits), 13.16% (10 splits), and 19.73% on TinyImageNet. Those results verify the orthogonality of our proposed method to regularization-based and replay-based CL methods, especially to the parameter regularization-based methods. Furthermore, we observe that the performance improvements brought by *ScrollNet* in the task-agnostic setting are comparatively less pronounced on TinyImageNet than on CIFAR100. This could be attributed to the limited capacity of ResNet18 for handling larger datasets, thereby constraining the effectiveness of *ScrollNet*.

Regarding the task-aware setting, our method also outperforms most of the baselines, except for LwF on CIFAR100 with 10 splits, BiC on TinyImageNet with 20 splits, and LUCIR on TinyImageNet with 5/10 splits. One possible reason for this is that the accuracies of these baselines in the task-aware setting are already quite high, leaving limited room for further improvement. Additionally, we find that the benefits of combining *ScrollNet* and LwF are not evident in other cases within the task-aware setting. This suggests that the advantage of incorporating *ScrollNet* with LwF in terms of cross-task representation learning may diminish in the task-aware setting. As for BiC and LUCIR, similar to the previous discussion, the limited capacity of ResNet18 on TinyImageNet also constrains the performance of our method.

Comparing the performance of the two variants of *ScrollNet*, we find that ScrollNet-4 consistently outperforms ScrollNet-2 in most cases. The reason behind this is that more model splitting corresponds to a finer weight ranking, providing a more precise "scrolling" mechanism that achieves a better stability-plasticity tradeoff. This holds particularly true for longer sequences of tasks, such as 20 splits, where *ScrollNet* with more model splitting yields superior results.

5. Discussion and future works

We believe that our work presents a novel direction for the continual learning community, as it introduces the pre-assignment of weight importance ranking before learning each task using a dynamic network, and provides an explicit way to strike a balance between stability and plasticity. The experiments have demonstrated the effectiveness of our proposed method. However, it is important to note that in this work, we have just implemented a simple dynamic network, i.e., considering only channel-wise model splitting and up to four splits throughout the network. Furthermore, our strategy (i.e., "scrolling") to leverage weight ranking knowledge for improving the performance of various CL methods is straightforward. Moving forward, we plan to explore more refined splitting techniques borrowing from the field of dynamic networks, and investigate alternative strategies for utilizing this prior knowledge. For example,

[1] https://github.com/mmasana/FACIL

Method	Exemplar	Task-agnostic			Task-aware		
		5 splits	10 splits	20 splits	5 splits	10 splits	20 splits
FT	No	25.47 ± 0.35	15.17 ± 1.06	7.27 ± 0.67	54.23 ± 1.46	49.97 ± 2.35	42.67 ± 4.15
ScrollNet-2	No	28.70 ± 0.56	16.60 ± 0.26	8.13 ± 0.31	58.17 ± 1.00	53.00 ± 1.56	49.70 ± 1.84
ScrollNet-4	No	$\mathbf{31.67 \pm 1.59}$	$\mathbf{18.66 \pm 0.49}$	$\mathbf{9.33 \pm 0.95}$	$\mathbf{60.47 \pm 1.53}$	$\mathbf{56.60 \pm 1.35}$	$\mathbf{55.27 \pm 2.11}$
LwF	No	43.80 ± 0.35	30.46 ± 1.74	17.90 ± 0.62	78.83 ± 0.76	$\mathbf{80.97 \pm 0.91}$	80.30 ± 0.70
ScrollNet-2	No	45.67 ± 0.67	30.63 ± 2.76	17.93 ± 1.27	$\mathbf{79.77 \pm 0.72}$	80.40 ± 0.85	79.47 ± 0.90
ScrollNet-4	No	$\mathbf{46.70 \pm 0.60}$	$\mathbf{32.37 \pm 0.76}$	$\mathbf{19.60 \pm 0.56}$	79.30 ± 0.72	79.93 ± 0.96	$\mathbf{81.47 \pm 1.06}$
EWC	No	31.57 ± 0.72	19.56 ± 0.65	9.80 ± 1.40	61.43 ± 0.64	57.47 ± 2.54	51.33 ± 1.70
ScrollNet-2	No	39.47 ± 0.76	26.57 ± 1.27	13.23 ± 1.27	69.03 ± 0.40	67.00 ± 2.57	62.80 ± 0.88
ScrollNet-4	No	$\mathbf{40.97 \pm 0.75}$	$\mathbf{28.03 \pm 1.88}$	$\mathbf{15.41 \pm 1.85}$	$\mathbf{71.47 \pm 0.85}$	$\mathbf{69.87 \pm 2.15}$	$\mathbf{67.97 \pm 1.63}$
MAS	No	35.23 ± 0.49	21.50 ± 1.48	9.70 ± 0.98	65.03 ± 0.21	61.53 ± 2.65	55.20 ± 2.26
ScrollNet-2	No	39.83 ± 0.90	25.93 ± 2.21	13.80 ± 1.85	69.03 ± 0.31	68.33 ± 2.79	64.73 ± 0.65
ScrollNet-4	No	$\mathbf{40.04 \pm 1.11}$	$\mathbf{27.40 \pm 0.52}$	$\mathbf{15.50 \pm 1.32}$	$\mathbf{70.40 \pm 1.14}$	$\mathbf{69.83 \pm 1.06}$	$\mathbf{69.97 \pm 1.02}$
iCaRL	2000	53.76 ± 0.55	42.83 ± 2.38	32.57 ± 1.80	77.07 ± 0.59	78.97 ± 2.15	81.10 ± 1.39
ScrollNet-2	2000	$\mathbf{54.90 \pm 0.46}$	$\mathbf{45.30 \pm 2.21}$	35.93 ± 1.60	77.87 ± 0.61	80.73 ± 1.75	82.97 ± 1.01
ScrollNet-4	2000	54.80 ± 0.74	44.87 ± 1.17	$\mathbf{37.03 \pm 1.53}$	$\mathbf{77.93 \pm 0.35}$	$\mathbf{81.17 \pm 1.04}$	$\mathbf{83.40 \pm 1.56}$
BiC	2000	58.20 ± 0.62	49.26 ± 1.05	37.70 ± 1.51	80.60 ± 0.26	83.43 ± 1.17	85.40 ± 0.62
ScrollNet-2	2000	58.50 ± 0.36	49.67 ± 1.44	$\mathbf{39.00 \pm 1.44}$	$\mathbf{81.27 \pm 0.42}$	$\mathbf{83.60 \pm 1.45}$	$\mathbf{85.97 \pm 0.68}$
ScrollNet-4	2000	$\mathbf{58.77 \pm 0.87}$	$\mathbf{49.72 \pm 1.73}$	38.64 ± 0.47	81.00 ± 0.47	83.47 ± 1.50	85.73 ± 0.57
LUCIR	2000	54.80 ± 0.82	41.97 ± 1.80	34.23 ± 0.51	81.03 ± 0.15	83.23 ± 1.50	85.37 ± 0.76
ScrollNet-2	2000	54.50 ± 0.78	42.90 ± 1.41	34.83 ± 1.76	80.87 ± 0.57	83.43 ± 1.29	85.17 ± 1.07
ScrollNet-4	2000	$\mathbf{55.53 \pm 0.82}$	$\mathbf{45.46 \pm 0.58}$	$\mathbf{37.23 \pm 1.00}$	$\mathbf{81.60 \pm 0.69}$	$\mathbf{84.13 \pm 1.10}$	$\mathbf{86.10 \pm 1.05}$

Table 1. Average accuracy after the last task for various continual learning methods and their combination with ScrollNet-N (N is the number of sub-networks). We run experiments three times with random class orders on CIFAR100 and report averages $\pm$ deviations.

Method	Exemplar	Task-agnostic			Task-aware		
		5 splits	10 splits	20 splits	5 splits	10 splits	20 splits
FT	No	18.60 ± 0.46	10.70 ± 0.30	5.80 ± 0.62	37.20 ± 1.13	31.93 ± 1.62	28.67 ± 1.18
ScrollNet-2	No	$\mathbf{20.77 \pm 1.21}$	$\mathbf{12.60 \pm 0.61}$	6.97 ± 0.68	41.83 ± 1.50	38.67 ± 1.25	35.83 ± 1.50
ScrollNet-4	No	20.50 ± 0.56	12.37 ± 0.61	$\mathbf{7.63 \pm 0.81}$	$\mathbf{43.60 \pm 0.75}$	$\mathbf{40.03 \pm 0.70}$	$\mathbf{40.07 \pm 1.47}$
LwF	No	34.67 ± 0.76	24.23 ± 1.66	15.83 ± 1.22	65.87 ± 0.29	67.93 ± 1.10	68.03 ± 1.12
ScrollNet-2	No	35.76 ± 0.84	24.80 ± 1.74	15.33 ± 1.38	$\mathbf{66.40 \pm 0.61}$	$\mathbf{68.03 \pm 0.90}$	67.27 ± 1.23
ScrollNet-4	No	$\mathbf{36.47 \pm 0.93}$	$\mathbf{24.88 \pm 1.17}$	$\mathbf{18.33 \pm 1.40}$	65.70 ± 0.00	66.97 ± 1.12	$\mathbf{72.13 \pm 0.56}$
EWC	No	29.37 ± 0.90	19.37 ± 1.08	9.87 ± 0.42	54.03 ± 1.70	51.07 ± 0.38	44.27 ± 1.21
ScrollNet-2	No	$\mathbf{32.00 \pm 0.10}$	24.70 ± 0.98	15.17 ± 0.21	$\mathbf{60.73 \pm 0.83}$	61.27 ± 0.64	60.40 ± 2.10
ScrollNet-4	No	30.87 ± 1.46	$\mathbf{25.23 \pm 0.38}$	$\mathbf{17.23 \pm 1.02}$	59.70 ± 2.04	$\mathbf{64.23 \pm 0.40}$	$\mathbf{64.00 \pm 1.68}$
MAS	No	27.57 ± 0.90	16.43 ± 0.64	9.40 ± 0.35	51.57 ± 1.51	47.70 ± 0.46	45.40 ± 1.90
ScrollNet-2	No	$\mathbf{31.70 \pm 0.36}$	21.70 ± 0.62	12.77 ± 0.42	58.03 ± 0.67	57.57 ± 0.60	55.63 ± 1.63
ScrollNet-4	No	30.57 ± 1.40	$\mathbf{24.30 \pm 0.56}$	$\mathbf{15.63 \pm 0.86}$	$\mathbf{58.30 \pm 2.00}$	$\mathbf{60.67 \pm 0.42}$	$\mathbf{62.03 \pm 1.89}$
iCaRL	2000	37.10 ± 0.70	31.50 ± 0.78	21.13 ± 0.68	60.57 ± 0.93	67.50 ± 1.21	68.57 ± 1.01
ScrollNet-2	2000	$\mathbf{39.40 \pm 0.87}$	32.6 ± 1.06	23.17 ± 1.45	$\mathbf{61.80 \pm 0.70}$	68.37 ± 1.46	70.3 ± 1.65
ScrollNet-4	2000	38.93 ± 0.91	$\mathbf{33.97 \pm 1.11}$	$\mathbf{25.33 \pm 1.24}$	61.63 ± 0.67	$\mathbf{69.10 \pm 1.04}$	$\mathbf{71.17 \pm 1.10}$
BiC	2000	43.70 ± 0.72	36.03 ± 0.50	26.03 ± 1.65	66.73 ± 0.92	71.80 ± 1.13	$\mathbf{76.03 \pm 1.12}$
ScrollNet-2	2000	45.00 ± 0.40	36.93 ± 0.55	26.87 ± 1.10	$\mathbf{68.17 \pm 0.50}$	72.27 ± 0.83	75.73 ± 0.66
ScrollNet-4	2000	$\mathbf{45.10 \pm 1.05}$	$\mathbf{37.17 \pm 0.29}$	$\mathbf{27.30 \pm 1.21}$	67.93 ± 0.15	$\mathbf{72.33 \pm 0.96}$	75.50 ± 0.82
LUCIR	2000	$\mathbf{36.97 \pm 1.06}$	24.67 ± 1.00	17.73 ± 1.46	$\mathbf{66.70 \pm 0.92}$	$\mathbf{70.30 \pm 0.44}$	73.33 ± 0.40
ScrollNet-2	2000	35.23 ± 0.29	23.90 ± 1.71	16.83 ± 1.25	65.53 ± 0.78	68.93 ± 1.50	72.33 ± 0.51
ScrollNet-4	2000	34.43 ± 0.67	$\mathbf{24.77 \pm 1.50}$	$\mathbf{18.17 \pm 0.81}$	66.17 ± 0.60	70.27 ± 0.85	$\mathbf{74.53 \pm 0.60}$

Table 2. Average accuracy after the last task for various continual learning methods and their combination with ScrollNet-N (N is the number of sub-networks). We run experiments three times with random class orders on TinyImageNet and report averages $\pm$ deviations.

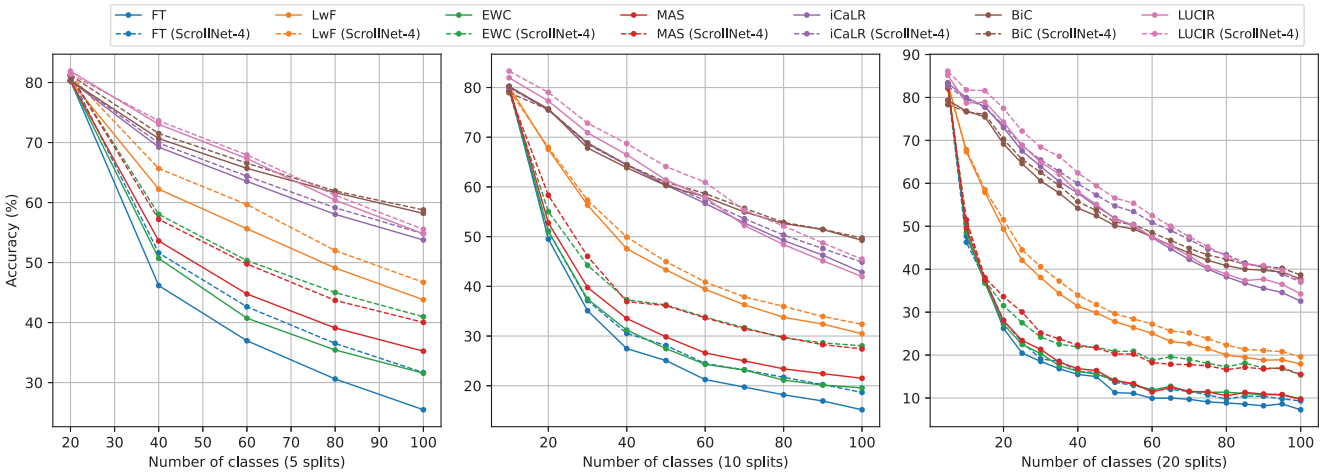

Figure 4. Results on CIFAR100 dataset (**task-agnostic**). We report the average accuracies of three runs after each task, each with a random class order.

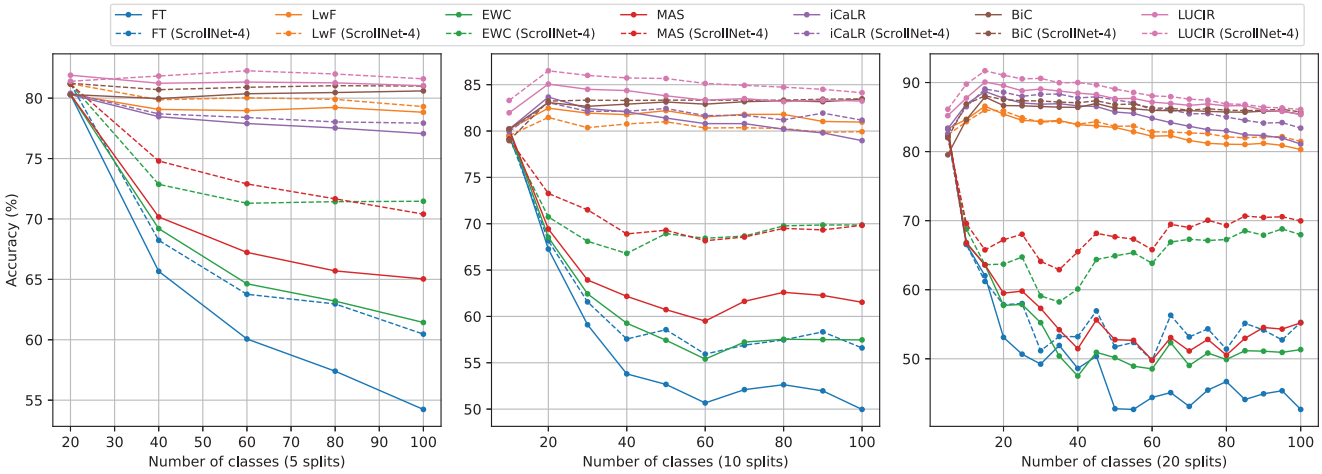

Figure 5. Results on CIFAR100 dataset (**task-aware**). We report the average accuracies of three runs after each task, each with a random class order.

one potential approach could involve calculating the step size of 'scrolling' based on the correlation between each task. Given that the performance improvement on TinyImageNet is not particularly evident, we are also planning to explore the effectiveness of our method when using larger models, such as ResNet101 and ViTs [11].

6. Conclusion

In this paper, we introduce ScrollNet, a scrolling neural network designed for continual learning. ScrollNet functions as a dynamic network that assigns the ranking of weight importance for each task before data exposure, thereby achieving a more favorable tradeoff between stability and plasticity during sequential task learning by adjusting this ranking for different tasks. Furthermore, we demonstrate that ScrollNet can be combined with various continual learning methods, including regularization-based and replay-based approaches. We validate the effectiveness of our proposed method through experiments conducted on CIFAR100 and TinyImagenet datasets.

7. Limitation

The training time will increase when combined with our proposed *ScrollNet* due to multiple forward passes with different sub-networks during training. This is especially true when *ScrollNet* has a larger number of model splittings. However, during inference, the computational cost of combining *ScrollNet* will remain the same because only the whole model will be executed.

Acknowledgements

We acknowledge the support from the Spanish Government funding for projects PID2022-143257NB-I00, TED2021-132513B-I00.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3931–3940, 2020.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017.

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[6] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2018.

[7] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[8] Kyunghyun Cho and Yoshua Bengio. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. *arXiv preprint arXiv:1406.7362*, 2014.

[9] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE TPAMI*, 2021.

[10] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE TPAMI*, 2021.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4040–4050, 2021.

[13] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

[14] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[15] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[16] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021.

[17] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *ECCV*, pages 466–483. Springer, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019.

[20] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[21] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[22] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR, 2019.

[23] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[24] Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. In *European Conference on Computer Vision*, pages 219–236. Springer, 2022.

[25] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[27] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[28] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[29] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective exe-

cution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[30] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *CVPR*, pages 226–227, 2020.

[31] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[32] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[33] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[34] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[35] Marc Masana, Tinne Tuytelaars, and Joost van de Weijer. Ternary feature masks: continual learning without any forgetting. *2nd CLVISION workshop in CVPR 2021*, 2020.

[36] Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

[37] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[38] Nikhil Mehta, Kevin Liang, Vinay Kumar Verma, and Lawrence Carin. Continual learning using a bayesian nonparametric dictionary of weight factors. In *International Conference on Artificial Intelligence and Statistics*, pages 100–108. PMLR, 2021.

[39] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320, 2020.

[40] Augustus Odena, Dieterich Lawson, and Christopher Olah. Changing model behavior at test-time using reinforcement learning. *arXiv preprint arXiv:1702.07780*, 2017.

[41] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[42] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[43] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2018.

[44] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[45] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[46] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NeurIPS*, pages 2994–3003, 2017.

[47] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 9634–9643, 2021.

[48] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[49] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[50] Eli Verwimp, Kuo Yang, Sarah Parisot, Lanqing Hong, Steven McDonagh, Eduardo Pérez-Pellitero, Matthias De Lange, and Tinne Tuytelaars. Clad: A realistic continual learning benchmark for autonomous driving. *Neural Networks*, 161:659–669, 2023.

[51] Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[52] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021.

[53] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[54] Maciej Wołczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzcinski. Zero time waste: Recycling predictions in early exit neural networks. *Advances in Neural Information Processing Systems*, 34:2516–2528, 2021.

[55] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

[56] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learn-

ing to generate new categories without forgetting. *NeurIPS*, 31:5962–5972, 2018.

[57] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019.

[58] Fei Yang, Luis Herranz, Yongmei Cheng, and Mikhail G Mozerov. Slimmable compressive autoencoders for practical neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4998–5007, 2021.

[59] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2018.

[60] Lu Yu, Xialei Liu, and Joost Van de Weijer. Self-training for class-incremental semantic segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[61] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. pages 3987–3995. PMLR, 2017.