DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion

Matthieu Cord, Arthur Douillard, Alexandre Ramé, Guillaume Couairon

DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion

Arthur Douillard[1,2], Alexandre Ramé[1], Guillaume Couairon[1,3], Matthieu Cord[1,4]
[1]Sorbonne Université, [2]Heuritech, [3]Meta AI, [4]valeo.ai
arthur.douillard@heuritech.com, {alexandre.rame, matthieu.cord}@sorbonne-universite.fr,
gcouairon@fb.com

Abstract

Deep network architectures struggle to continually learn new tasks without forgetting the previous tasks. A recent trend indicates that dynamic architectures based on an expansion of the parameters can reduce catastrophic forgetting efficiently in continual learning. However, existing approaches often require a task identifier at test-time, need complex tuning to balance the growing number of parameters, and barely share any information across tasks. As a result, they struggle to scale to a large number of tasks without significant overhead.

In this paper, we propose a transformer architecture based on a dedicated encoder/decoder framework. Critically, the encoder and decoder are shared among all tasks. Through a dynamic expansion of special tokens, we specialize each forward of our decoder network on a task distribution. Our strategy scales to a large number of tasks while having negligible memory and time overheads due to strict control of the expansion of the parameters. Moreover, this efficient strategy doesn't need any hyperparameter tuning to control the network's expansion. Our model reaches excellent results on CIFAR100 and state-of-the-art performances on the large-scale ImageNet100 and ImageNet1000 while having fewer parameters than concurrent dynamic frameworks.[1]

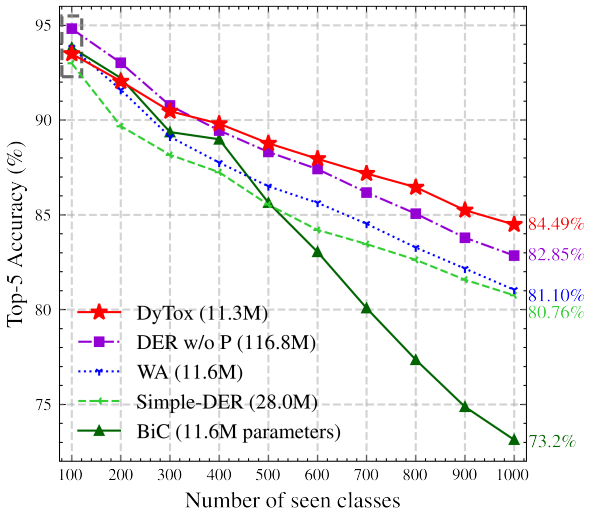

Figure 1: **DyTox's continual learning performance on ImageNet1000**: for each task, 100 new classes are learned while previously learned classes are not fully accessible but shouldn't be forgotten. Our strategy DyTox (in **red**) is state-of-the-art by a large margin. Note that at the initial step before the continual process begins (denoted by a dashed rectangle ⌜⌟), our model has performance comparable to other baselines: the performance gain is achieved by reducing catastrophic forgetting. Moreover, we have systematically fewer parameters than previous approaches.

1. Introduction

Most of the deep learning literature focuses on learning a model on a fixed dataset. However, real-world data constantly evolve through time, leading to ever-changing distributions: *i.e.*, new classes or domains appeared. When a model loses access to previous classes data (*e.g.*, for privacy reasons) and is fine-tuned on new classes data, it **catastrophically forgets** the old distribution. Continual learning models aim at balancing a rigidity/plasticity trade-off where old data are not forgotten (rigidity to changes) while learning new incoming data (plasticity to adapt). Despite recent advances, it is still an open challenge.

A growing amount of efforts have emerged to tackle catastrophic forgetting [49, 34, 63, 29, 18, 64]. Recent works [65, 39, 30, 21, 24, 54] dynamically expand the network architectures [65, 39] or re-arrange their internal structures [21, 54, 30, 24]. Unfortunately at test-time, they require to know the task to which the test sample belongs — in order to know which parameters should be used. More recently, DER [64] and Simple-DER [41] discarded the need for this task identifier by learning a single classifier on the concatenation of all produced embeddings by different sub-

[1]Code is released at https://github.com/arthurdouillard/dytox

sets of parameters. Yet, these strategies induce dramatic memory overhead when tackling a large number of tasks, and thus need complex pruning as post-processing.

To improve the ease of use of continual learning frameworks for real-world applications, we aim to design a dynamically expandable representation (almost) 'for free' by having the three following properties: #1 **limited memory overhead** as the number of tasks grows, #2 **limited time overhead** at test time and #3 **no setting-specific hyperparameters** for improved robustness when faced to an unknown (potentially large) number of tasks.

To this end, we leverage the computer vision transformer ViT [15]. Transformers [60] offer a very interesting framework to satisfy the previously mentioned constraints. Indeed, we build upon this architecture to design a **encoder/decoder strategy**: the encoder layers are shared among all members of our dynamic network; the unique decoder layer is also shared but its forward pass is specialized by a **task-specific learned token** to produce task-specific embeddings. Thus, the memory growth of the dynamic network is extremely limited: only a 384d vector per task, validating property #1. Moreover, this requires no hyperparameter tuning (property #3). Finally, the decoder is explicitly designed to be computationally lightweight (satisfying property #2). We nicknamed our framework, DyTox, for **DYnamic TOken eXpansion**. To the best of our knowledge, we are the first to apply the transformer architecture to continual computer vision.

Our strategy is robust to different settings, and can easily scale to a large number of tasks. In particular, we validate the efficiency of our approach on CIFAR100, ImageNet100, and ImageNet1000 (displayed on Fig. 1) for multiple settings. We reach state-of-the-art results, with only a small overhead thanks to our efficient dynamic strategy.

2. Related work

Continual learning models tackle the catastrophic forgetting of the old classes [56, 22]. In computer vision, most of continual learning strategies applied on large-scale datasets use rehearsal learning: a limited amount of the training data of old classes is kept during training [50]. This data is usually kept in raw form (*e.g.*, pixels) [49, 4, 9] but can also be compressed [26, 31], or trimmed [17] to reduce memory overhead; others store only a model to generate new samples of past classes [33, 55, 38]. In addition, most approaches aim at limiting the changes in the model when new classes are learned. These constraints can be directly applied on the weights [34, 66, 1, 7], intermediary features [29, 14, 69, 18, 16], prediction probabilities [40, 49, 4, 5], or on the gradients [43, 8, 20, 52]. All these constraint-based methods use the same static network architectures which doesn't evolve through time, usually a ResNet [27], a LeNet [36], or a small MLP.

Continual dynamic networks In contrast, our paper and others focus on designing **dynamic architectures** that best handle a growing training distribution [65, 39], in particular by dynamically creating (sub-)members each specialized in one specific task [21, 24, 30, 51, 10, 61]. Unfortunately, previous approaches often require the sample's task identifier at test-time to select the right subset of parameters. We argue this is an unrealistic assumption in a real-life situation where new samples could come from any task. Recently, DER [64] proposed a dynamic expansion of the representation by adding a new feature extractor per task. All extractors' embeddings would then be concatenated and fed to a unified classifier, discarding the need for a task identifier at test-time. To limit an explosion in the number of parameters, they aggressively prune each model after each task using the HAT [54] procedure. Unfortunately, the pruning is hyperparameter sensitive. Therefore, hyperparameters are tuned differently on each experiment: for example, learning a dataset in 10 steps or in 50 steps use different hyperparameters. While being impracticable, it is also unrealistic because the number of classes is not known in advance in a true continual situation. Simple-DER [41] also uses multiple extractors, but its pruning method doesn't need any hyperparameters; the negative counterpart is that Simple-DER controls less the parameter growth (2.5x higher than a base model). In contrast, we propose a framework dedicated to continual learning that seamlessly enables a task-dynamic strategy, efficient on all settings, without any setting-dependant modification and at almost no memory overhead. We share early class-agnostic [45] layers similarly to TreeNets [37] and base our strategy on the Transformer architecture.

Transformers were first introduced for machine translation [60], with the now famous self-attention. While the original transformer was made of encoder and decoder layers, later transformers starting from BERT [13] used a succession of identical encoder blocks. Then, ViT [15] proposed to apply transformers to computer vision by using patches of pixels as tokens. Multiple recent works, including DeiT [58], CaiT [59], ConVit [11], and Swin [42], improved ViT with architecture and training procedures modifications. PerceiverIO [32] proposed a general architecture whose output is adapted to different modalities using specific learned tokens, and whose computation is reduced using a small number of latent tokens. Despite being successful across various benchmarks, transformers have not yet been considered for continual computer vision to the best of our knowledge. Yet, we don't use the transformer architecture for its own sake, but rather because of the intrinsic properties of transformers; in particular, the seminal encoder/decoder framework allows us to build an efficient architecture with strong capabilities against catastrophic forgetting.

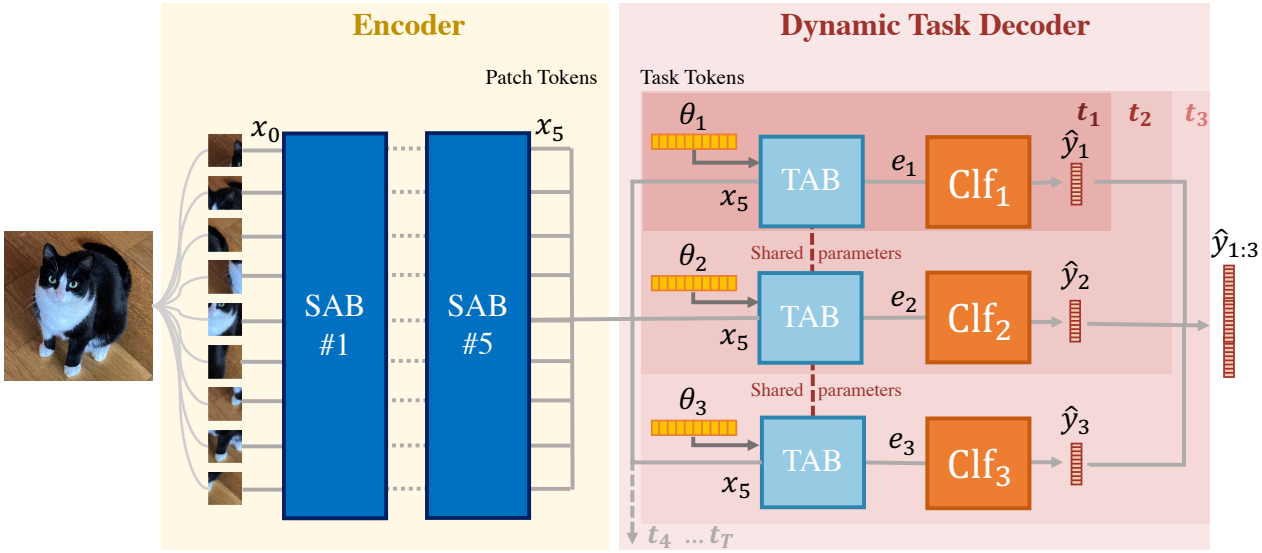

Figure 2: **DyTox transformer model**. An image is first split into multiple patches, embedded with a linear projection. The resulting patch tokens are processed by 5 successive Self-Attention Blocks (SAB) (Sec. 3.1). For each task ($t = 1 \dots T$), the processed patch tokens are then given to the Task-Attention Block (TAB) (Sec. 3.2): each forward through the TAB is modified by a different task-specialized token $\boldsymbol{\theta}_t$ for $t \in \{1 \dots T\}$ (Sec. 3.3). The T final embeddings are finally given separately to independent classifiers Clf_t each predicting their task's classes C^t. All $|C^{1:T}|$ logits are activated with a sigmoid. For example, at task $t = 3$, one forward is done through the SABs and three task-specific forwards through the unique TAB.

3. DyTox transformer model

Our goal is to learn a unified model that will classify an increasingly growing number of classes, introduced in a fixed amount of steps T. At a given step $t \in \{1 \dots T\}$, the model is exposed to new data belonging to new classes. Specifically, it learns from samples $\{(x_i^t, y_i^t)\}_i$, where x_i^t is the i-th image of this task t and y_i^t is the associated label within the label set $\mathcal{C}^t$. All task label sets are exclusive: $\mathcal{C}^0 \cap \mathcal{C}^1 \dots \mathcal{C}^T = \emptyset$. The main challenge is that the data are fully available only temporarily: following most previous works, only a few samples from previous tasks $\{1 \dots t-1\}$ are available for training at step t as rehearsing data. Yet, the model should remain able to classify test data coming from all seen classes $\mathcal{C}^{1:t}$. A table of notations is provided in the supplementary materials.

The Fig. 2 displays our DyTox framework, which is made of several components (SAB, TAB, and Task Tokens) that we describe in the following sections.

3.1. Background

The vision transformer [15] has three main components: the patch tokenizer, the encoder made of Self-Attention Blocks, and the classifier.

Patch tokenizer The fixed-size input RGB image is cropped into N patches of equal dimensions and then projected with a linear layer to a dimension D. Both operations, the cropping and projection, are done with a single 2D convolution whose kernel size is equal to its stride size. The resulting tensor $x_0 \in \mathbb{R}^{N \times D}$ is extended with a learned class token $x_{\text{cls}} \in \mathbb{R}^D$ resulting in a tensor of shape $\mathbb{R}^{(N+1) \times D}$. Following [23], a learned positional embedding $p \in \mathbb{R}^{(N+1) \times D}$ is added (element-wise).

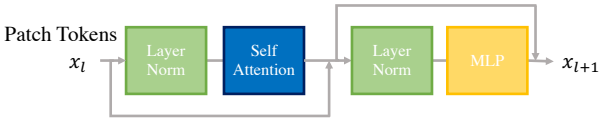

Figure 3: **The Self-Attention Block (SAB)** combines a Self-Attention (SA), two Layer Norms, and one MLP with a single hidden layer. As in a ResNet, two shortcuts are used with element-wise addition.

Self-Attention (SA) based encoder The tokens are fed to a stack of transformer blocks that we denote here as Self-Attention Blocks (SABs):

$$\begin{aligned} x'_l &= x_l + \text{SA}_l\left(\text{Norm}_{l,1}\left(x_l\right)\right), \\ x_{l+1} &= x'_l + \text{MLP}_l\left(\text{Norm}_{l,2}\left(x'_l\right)\right), \end{aligned} \tag{1}$$

with SA a Self-Attention layer [60], Norm a layer normalization [2], and MLP a Multi-Layer Perceptron with a single hidden layer. We repeat these operations for each SAB, from $l = 1$ to $l = L$. The resulting tensor (which keeps the same dimension after every block) is $x_L \in \mathbb{R}^{(N+1) \times D}$. We display a visual illustration of a SA Block in Fig. 3.

Classifier In the original vision transformer (ViT [15]), a learned vector called the "*class token*" is appended to the patch tokens after the tokenizer. This special class token, when processed after all the SABs, is given to a linear classifier with a softmax activation to predict the final probabilities. However, more recent works, as CaiT [59], propose instead to introduce the class token only at the ultimate or penultimate SAB to improve classification performance.

3.2. Task-Attention Block (TAB)

Contrary to previous transformer architectures, we don't have a class token, but rather what we nicknamed "**task tokens**"; the learned token of the i^{th} task is denoted $\boldsymbol{\theta}_i$. This special token will only be added at the last block. To exploit this task token, we define a new attention layer, that we call the Task-Attention. It first concatenates the patch tokens x_L produced by the ultimate SAB with a task token $\boldsymbol{\theta}_i$:

$$z_i = [\boldsymbol{\theta}_i, x_L] \in \mathbb{R}^{(\mathbb{N}+1)\times\mathbb{D}} . \tag{2}$$

This is then given to the Task-Attention (TA), inspired by the Class-Attention of Touvron et al. [59]:

$$\begin{aligned} Q_i &= W_q \boldsymbol{\theta}_i \,, \\ K_i &= W_k z_i \,, \\ V_i &= W_v z_i \,, \\ A_i &= \text{Softmax}\left(Q_i \cdot K_i^T / \sqrt{d/h}\right) \,, \\ O_i &= W_o A_i V_i + b_o \in \mathbb{R}^{1\times\mathbb{D}} \,, \end{aligned} \tag{3}$$

with d being the embedding dimension, and h the number of attention heads [60]. Contrary to the classical Self-Attention, the Task-Attention defines its query (Q_i) only from the task-token $\boldsymbol{\theta}_i$ without using the patch tokens x_L. The Task-Attention Block (TAB) is then a variation of the SAB where the attention is a Task-Attention (TA):

$$\begin{aligned} c' &= c + \text{TA}\left(\text{Norm}_1\left(z\right)\right) \,, \\ c'' &= c' + \text{MLP}\left(\text{Norm}_2\left(c'\right)\right) \,. \end{aligned} \tag{4}$$

Overall, our new architecture can be summarized by the repetition of SA Blocks $\{\text{SAB}_l\}_{l=1}^{L}$ (defined in Eq. 1) ended by a single TA Block TAB (defined in Eq. 4):

$$e_i = \text{TAB} \circ ([\boldsymbol{\theta}_i, \text{SAB}_{l=L} \circ \ldots \text{SAB}_{l=1}(x_0)]) \in \mathbb{R}^D \,. \tag{5}$$

The final embedding e_i is fed to a classifier clf made of a Norm_c and a linear projection parametrized by $\{W_c, b_c\}$:

$$\tilde{y}_i = \text{Clf}(e_i) = W_c \,\text{Norm}_c(e_i) + b_c \,. \tag{6}$$

3.3. Dynamic task token expansion

We defined in the previous section our base network, made of a succession of SABs and ended by a single TAB. As detailed, the TAB has two inputs: the patch tokens x_L extracted from the image and a learned task-token $\boldsymbol{\theta}_i$. We'll now detail how our framework evolves in a continual situation at each new step.

During the first step, there is only one task token $\boldsymbol{\theta}_1$. At each new step, we propose to expand our parameter space by creating a new task token while keeping the previous ones. Thus, after t steps, we have t task tokens ($\boldsymbol{\theta}_i$ for $i \in \{1 \ldots t\}$). Given an image x — belonging to any of the seen tasks $\{1 \ldots t\}$ — our model tokenizes it into x_0, and processes it through the multiple SABs: this outputs the patch tokens x_L. Finally, our framework does as many forward passes through the TAB as there are tasks: critically, each TAB forward passes is executed with a different task token $\boldsymbol{\theta}_i$, resulting in different task-specific forwards, each producing the task-specific embeddings e_i (see Fig. 2):

$$\begin{aligned} e_1 &= \text{TAB}([\boldsymbol{\theta}_1, x_L]) \,, \\ e_2 &= \text{TAB}([\boldsymbol{\theta}_2, x_L]) \,, \\ &\ldots \\ e_t &= \text{TAB}([\boldsymbol{\theta}_t, x_L]) \,. \end{aligned} \tag{7}$$

Rather than concatenating all embeddings $\{e_1, e_2, \ldots, e_t\}$ together and feeding them to one classifier, we leverage **task-specific classifiers**. Each classifier clf_i is made of a Norm_i and a linear projection parametrized by $\{W_i, b_i\}$, with $W_i \in \mathbb{R}^{\mathcal{C}^i \times D}$ and $b \in \mathbb{R}^{\mathcal{C}^i}$. It takes as input its task-specific embedding e_i and returns:

$$\hat{y}_i = \text{Clf}_i(e_i) = \sigma(W_i \,\text{Norm}_i \, e_i + b_i) \,, \tag{8}$$

the predictions for the classes $y_i \in \mathcal{C}^i$, where $\sigma(x) = {}^1/_{(1+e^{-x})}$ is the sigmoid activation. In comparison with the softmax activation, the element-wise sigmoid activation reduces the overconfidence in recent classes. Consequently, the model is better calibrated, which is an important attribute of continual model [3, 63, 68]. The loss is the binary-cross entropy. The independent classifiers paradigm coupled with the sigmoid activation and binary cross-entropy loss exclude explicitly a late fusion [48] of the task embeddings resulting in more **specialized classifiers**.

The overall structure of the DyTox strategy is illustrated in Fig. 2. We also show in Algo. 1 the pseudo-code of a forward pass at test-time after having learned the task t. Critically, the test image can belong to any of the previously seen tasks $\{1 \ldots t\}$. Our dynamic task token expansion is more efficient than a naive parameter expansion that would create a new copy of the whole network for each new task. (1) Our expansion is limited to a new task token per new task, which is only $d = 384$ new parameters. This is small compared to the total model size (≈ 11 million parameters). The **memory overhead is thus almost null**. (2) The computationally intensive blocks (*i.e.*, the SABs) are executed

Algorithm 1 DyTox's forward pass at step t

Input: x_0 (initial patch tokens), y (ground-truth labels)
Output: $\hat{y}_{1:t}$ (predictions for all classes of $\mathcal{C}^{1:t}$)

$x_L \leftarrow \text{SAB}_{l=L} \circ ... \text{SAB}_{l=1}(x_0)$ ▷ Sec. 3.1
for $i \leftarrow 1$; $i \leq t$; $i{+}{+}$ **do**
  $e_i \leftarrow \text{TAB}([\boldsymbol{\theta}_i, x_L])$ ▷ Sec. 3.2
  $\hat{y}_i \leftarrow \text{Clf}_i(e_i)$ ▷ Sec. 3.3
end for
$\hat{y}_{1:t} \leftarrow [\hat{y}_1, \ldots, \hat{y}_t]$

only once despite learning multiple tasks. In contrast, the TAB has as many forwards as there are tasks. Though, this induces minimal overhead because the **Task-Attention has a linear complexity w.r.t the number of patches** while the Self-Attention is quadratic. Therefore, the time overhead is sub-linear. We quantitatively show this in Sec. 4.

Context The current transformer paradigm starting from BERT [13] and continuing with ViT [15] is based on a encoder+classifier structure. Differently, our dynamic framework strays is a resurgence of the encoder/decoder structure of the original transformer [60]: the encoder is shared (both in memory and execution) for all outputs. The decoder parameters are also shared, but its execution is task-specific with each task token, with each forward akin to a task-specific expert chosen from a mixture of experts [44]. Moreover, multi-tasks text-based transformers have natural language tokens as an indicator of a task [46] (*e.g.* "summarize the following text"), in our context of vision we used our defined task tokens as indicators.

Losses Our model is trained with three losses: (1) the classification loss $\mathcal{L}_{\text{clf}}$, a binary-cross entropy, (2) a knowledge distillation [28] $\mathcal{L}_{\text{kd}}$ applied on the probabilities, and (3) the divergence loss $\mathcal{L}_{\text{div}}$. The distillation loss helps to reduce forgetting. It is arguably quite naive, and more complex distillation losses [53, 29, 18] could further improve results. The divergence loss, inspired from the "auxiliary classifier" of DER [64], uses the current last task's embedding e_t to predict $(|\mathcal{C}^t| + 1)$ probabilities: the current last task's classes $\mathcal{C}^t$ and an extra class representing all previous classes that can be encountered via rehearsal. This classifier is discarded at test-time and encourages a better diversity among task tokens. The total loss is:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{clf}} + \alpha\mathcal{L}_{\text{kd}} + \lambda\mathcal{L}_{\text{div}}\,, \tag{9}$$

with λ a hyperparameter set to 0.1 for **all** experiments. α correspond to the fraction of the number of old classes over the number of new classes $\frac{|C^{1:t-1}|}{|C^{1:t}|}$ as done by [68]. Therefore, α is automatically set; this removes the need to finely tune this hyperparameter.

Hyperparameter	CIFAR	ImageNet
# SAB	5	
# CAB	1	
# Attentions Heads	12	
Embed Dim	384	
Input Size	32	224
Patch Size	4	16

Table 1: **DyTox's architectures** for CIFAR and ImageNet. The only difference between the two architectures is the patch size, as the image sizes vary between datasets.

4. Experiments

4.1. Benchmarks & implementation

Benchmarks & Metrics We evaluate our model on CIFAR100 [35], ImageNet100 and ImageNet1000 [12] (descriptions in the supplementary materials) under different settings. The standard continual scenario in ImageNet has 10 steps: thus we add 10 new classes per step in ImageNet100, and 100 new classes per step in ImageNet1000. In CIFAR100, we compare performances on 10 steps (10 new classes per step), 20 steps (5 new classes per step), and 50 steps (2 new classes per step). In addition to the top-1 accuracy, we also compare the top-5 accuracy on ImageNet. We report the "*Avg*" accuracy which is the average of the accuracies after each step as defined by [49]. We also report the final accuracy after the last step ("*Last*"). Finally, in our tables, "*#P*" denotes the parameters count in million after the final step.

Implementation details As highlighted in Table 1, our network has the same structure across all tasks. Specifically, we use 5 Self-Attention Blocks (SABs), 1 Task-Attention Block (TAB). All 6 have an embedding dimension of 384 and 12 attention heads. We designed this shallow transformer to have a comparable parameters count to other baselines, but also made it wider than usual "tiny" models [15, 58, 59]. We tuned all hyperparameters for CIFAR100 with 10 steps on a validation set made of 10% of the training set, and then kept them fixed for all other settings, ImageNet included. The only difference between the two datasets is that ImageNet images are larger; thus the patch size is larger, and overall the base transformer has slightly more parameters on ImageNet than on CIFAR (11.00M vs 10.72M) because of a bigger positional embedding. We use the attention with spatial prior (introduced by ConViT [11]) for all SABs which allows training transformers on a small dataset (like CIFAR) without pretraining on large datasets or complex regularizations. Following previous works [49, 64], we use for all models (baselines included) 2,000 images of rehearsal memory for CIFAR100

Methods	ImageNet100 10 steps					ImageNet1000 10 steps				
	#P	top-1		top-5		#P	top-1		top-5	
		Avg	Last	Avg	Last		Avg	Last	Avg	Last
ResNet18 joint	11.22	-	-	-	95.10	11.68	-	-	-	89.27
Transf. joint	11.00	-	79.12	-	93.48	11.35	-	73.58	-	90.60
E2E [4]	11.22	-	-	89.92	80.29	11.68	-	-	72.09	52.29
Simple-DER [41]	-	-	-	-	-	28.00	66.63	59.24	85.62	80.76
iCaRL [49]	11.22	-	-	83.60	63.80	11.68	38.40	22.70	63.70	44.00
BiC [29]	11.22	-	-	90.60	84.40	11.68	-	-	84.00	73.20
WA [68]	11.22	-	-	91.00	84.10	11.68	65.67	55.60	86.60	81.10
RPSNet [47]		-	-	87.90	74.00	-	-	-	-	-
DER w/o P [64]	112.27	**77.18**	66.70	**93.23**	87.52	116.89	68.84	60.16	88.17	82.86
DER† [64]	-	76.12	66.06	92.79	88.38	-	66.73	58.62	87.08	81.89
DyTox	11.01	**77.15**	**69.10**	92.04	**87.98**	11.36	**71.29**	**63.34**	**88.59**	**84.49**

Table 2: **Results on ImageNet-100 and ImageNet-1000 datasets**, learned with 10 steps of respectively 10 and 100 new classes. E2E [4] and Simple-DER [41] results come from their respective papers, and used a different class ordering. Other results come from [64]. The † symbol means that [64] needed setting-sensitive hyperparameters. Moreover, its reported parameters count was an average over all steps ([64] reported 14.52M on ImageNet1000): the final parameters count (necessarily higher) was not available.

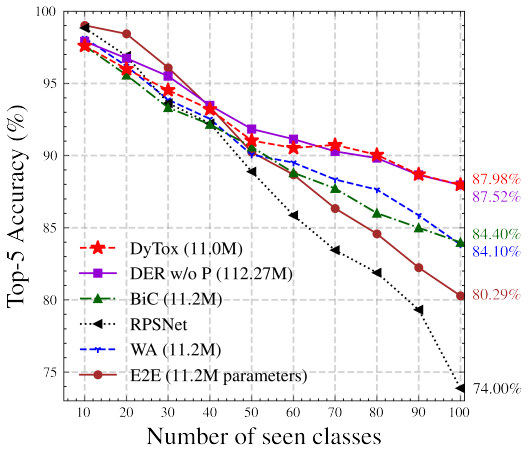

Figure 4: **Performance evolution on ImageNet100**. The top-5 accuracy (%) is reported after learning each task. Our model DyTox (in **red**) surpasses significantly most baselines, and is of equal performance as the complex DER that uses pruning with setting-specific hyperparameters.

and ImageNet100, and 20,000 images for ImageNet1000. The implementations of the continual scenarios are provided by Continuum [19]. Our network implementation is based on the DeiT [58] code base which itself uses extensively the timm library [62]. The code is released publicly[2]. The full implementation details are in the appendix.

[2]https://github.com/arthurdouillard/dytox

4.2. Quantitative results

ImageNet We report performances in Table 2 on the complex ImageNet dataset. The † marks the DER with setting-specific pruning, and DER w/o P is for the DER without pruning. In ImageNet100, DyTox reaches 69.10% and outperforms DER† by +3.04 percentage points (*p.p*) in "Last" top-1 accuracy. Though, DyTox and DER w/o P somehow perform similarly in "Avg" accuracy on this setup, as highlighted in the performance evolution displayed in Fig. 4. Most importantly, on the larger-scale ImageNet1000, DyTox systematically performs best on all metrics despite having lower parameters count. Specifically, DyTox reaches 71.29% in "Avg" top-1 accuracy, and 63.34% in "Last" top-1 accuracy. This outperforms the previous state-of-the-art DER w/o P (68.84% in "Avg", 60.16% in "Last") which has 10 ResNet18 in parallel and 116.89M parameters. Compared to the pruned DER†, DyTox has a +4.56 *p.p* in top-1 and a +1.51 *p.p* in top-5 for the "Avg" accuracy. All models evolutions on ImageNet1000 are illustrated in Fig. 1: DyTox constantly surpasses previous state-of-the-art models — despite having a comparable performance at the first step and fewer parameters.

DyTox is able to scale correctly while handling seamlessly the parameter growth by sharing most of the weights across tasks. In contrast, DER had to propose a complex pruning method; unfortunately, this pruning required different hyperparameter values for different settings. Despite this, the pruning in DER† is less efficient when classes diversity increase: DER† doubles in size between

Methods	10 steps #P	10 steps Avg	10 steps Last	20 steps #P	20 steps Avg	20 steps Last	50 steps #P	50 steps Avg	50 steps Last
ResNet18 Joint	11.22	-	80.41	11.22	-	81.49	11.22	-	81.74
Transf. Joint	10.72	-	76.12	10.72	-	76.12	10.72	-	76.12
iCaRL [49]	11.22	65.27 ± 1.02	50.74	11.22	61.20 ± 0.83	43.75	11.22	56.08 ± 0.83	36.62
UCIR [29]	11.22	58.66 ± 0.71	43.39	11.22	58.17 ± 0.30	40.63	11.22	56.86 ± 0.83	37.09
BiC [63]	11.22	68.80 ± 1.20	53.54	11.22	66.48 ± 0.32	47.02	11.22	62.09 ± 0.85	41.04
WA [68]	11.22	69.46 ± 0.29	53.78	11.22	67.33 ± 0.15	47.31	11.22	64.32 ± 0.28	42.14
PODNet [18]	11.22	58.03 ± 1.27	41.05	11.22	53.97 ± 0.85	35.02	11.22	51.19 ± 1.02	32.99
RPSNet [47]	56.5	68.60	57.05	-	-	-	-	-	-
DER w/o P [64]	112.27	75.36 ± 0.36	**65.22**	224.55	74.09 ± 0.33	62.48	561.39	72.41 ± 0.36	59.08
DER† [64]	-	74.64 ± 0.28	64.35	-	73.98 ± 0.36	**62.55**	-	72.05 ± 0.55	**59.76**
DyTox	10.73	73.66 ± 0.02	60.67 ± 0.34	10.74	72.27 ± 0.18	56.32 ± 0.61	10.77	70.20 ± 0.16	52.34 ± 0.26
DyTox+	10.73	**75.54** ± 0.10	62.06 ± 0.25	10.74	**75.04** ± 0.11	60.03 ± 0.45	10.77	**74.35** ± 0.05	57.09 ± 0.13

Table 3: **Results on CIFAR100** averaged over three different class orders. Baselines results are come from [64]. The † symbol means that [64] needed setting-sensitive hyperparameters. Moreover, its reported parameters count was an average over all steps: the final parameters count (necessarily higher) was not available.

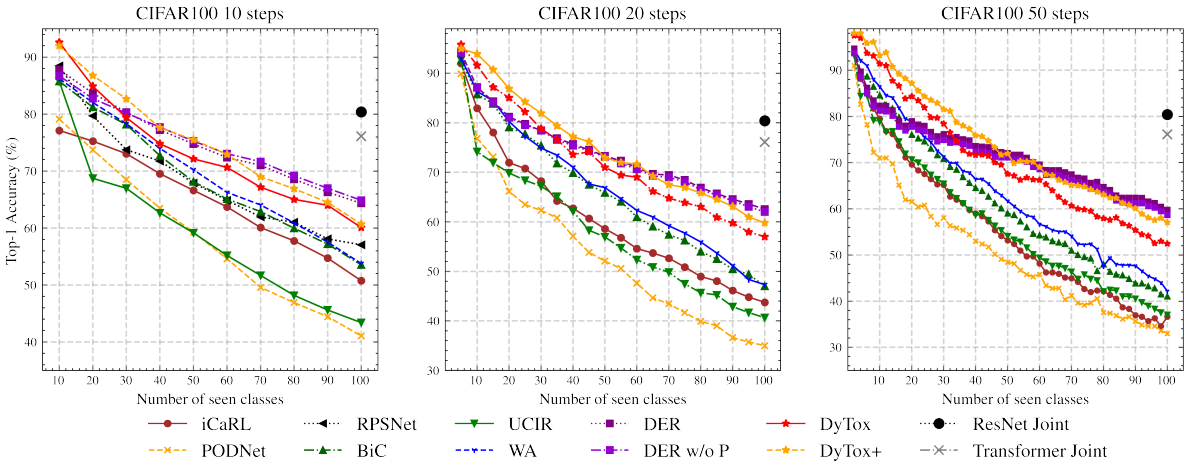

Figure 5: **Performance evolution on CIFAR100**. The top-1 accuracy (%) is reported after learning each task. **Left** is evaluated with 10 steps, **middle** with 20 steps, and **right** with 50 steps.

ImageNet100 and ImageNet1000 ([64] reports 7.67M *vs.* 14.52M) while handling the same amount of tasks (10). Note that these parameter counts reported for DER† in [64] are in fact averages over all steps: the final parameters count (necessarily higher) was not available and thus is not reported in our tables. Simple-DER also applies pruning but without hyperparameter tuning; while simpler, the pruning is also less efficient and induces larger model (28.00M parameters).

CIFAR100 Table 3 shows results for all approaches on CIFAR100. The more steps there are, the larger the forgetting is and thus the lower the performances are. Those settings are also displayed in Fig. 5 after each task. In every setting, DyTox is close to DER w/o P for much fewer parameters (up to 52x less). Critically, DyTox is significantly above other baselines: *e.g.* DyTox is up to +25% in "Last" accuracy in the 50 steps setup.

Improved training procedure To bridge the gap between DyTox and DER w/o P on CIFAR100, we introduce a new efficient training procedure for continual learning. Using MixUp [67], we linear interpolate new samples with existing samples. The interpolation factor $\lambda \sim \text{Beta}(\alpha, \alpha)$ is sampled with $\alpha = 0.8$: the pixels of two images are mixed ($x = \lambda x_1 + (1-\lambda)x_2$) as their labels ($y = \lambda y_1 + (1-\lambda)y_2$). MixUp was shown to have two main effects: (1) it diversifies the training images and thus enlarges the training distribution on the vicinity of each training sample [6] and (2) it improves the network calibration [25, 57], reducing the overconfidence in recent classes. Thus MixUp has shared motivation with the sigmoid activation. When Dy-

Training	1 step Last (↑)	50 steps Last (↑)	50 steps Forgetting (↓)
DyTox	76.12	52.34	33.15
DyTox+	77.51+1.39	57.09+4.75	31.50-1.65

Table 4: **"Last" accuracy and forgetting** [7] on CIFAR100 for the joint (1 step, no continual) and 50 steps settings.

Tox is combined with this MixUp procedure, nicknamed as DyTox+, this significantly improves the state-of-the-art in "Avg" accuracy in all three settings of Table 3. We also provide in the appendix further improvement for this new continual training procedure providing even larger gain on both CIFAR100 and ImageNet100.

4.3. Model introspection on CIFAR100

Memory overhead We only add a vector of size $d = 384$ per task; thus, the overhead in memory (not considering the growing classifier which is common for all continual models) is only of $+0.004\%$ per step. Even in the challenging setting of CIFAR100 with 50 tasks, our memory overhead is almost null ($+0.2\%$).

Computational overhead The vast majority of the computation is done in the SABs, thus shared among all tasks. The dynamical component of our model is located at the ultimate TAB. Moreover, the Task-Attention, contrary to the Self-Attention, has a time complexity linear in terms of tokens and not quadratic reducing the time overhead to an acceptable sub-linear amount. Overall, for each new task, one forward pass takes 2.24% more time than for the base transformer.

Training procedure introspection Our DyTox+ strategy with MixUp really reduces catastrophic forgetting and does not just improve raw performances. This is shown in Table 4, where we compare DyTox *vs.* DyTox+ strategies on CIFAR100. While MixUp only slightly improves by 1.39 *p.p* the accuracy in joint learning (no continual, 1 step), MixUp greatly improves the performance by 4.75 *p.p* in the 50 steps continual scenario. To further illustrate this, we also report the Chaudhry et al.'s forgetting [7] measure which compares how performances dropped compared to previous steps. MixUp reduces this forgetting by 1.65 *p.p* .

Model ablations We ablate the importance of the different components of DyTox in Table 5. We add on the base transformer a naive knowledge distillation [28] and a finetuning [4, 29, 18, 64] applied after each task on a balanced set of new data and rehearsal data. Finally, our DyTox strategy exploits directly the very nature of transformers (separated task information from the pixels information) to tackle catastrophic forgetting with three components: (1) a task token expansion, (2) a divergence classifier, and (3) independent classifiers. All three greatly improve over the baseline

		Knowledge Distillation	Finetuning	Token Expansion	Divergence Classifier	Indendepent Classifiers	Avg	Last
DyTox	Transformer						60.69	38.87
		✓					61.62	39.35
		✓	✓				63.42	42.21
	Dynamic	✓	✓	✓			67.30	47.57
		✓	✓	✓	✓		68.28	49.45
		✓	✓	✓	✓	✓	70.20	52.34

Table 5: **Ablations** of the different key components of our DyTox architecture. We report the average accuracy and the last accuracy on CIFAR100 for the setting with 50 steps.

transformer ($42.21\% \rightarrow 52.34\%$ in "Last") while having almost no memory overhead ($+0.2\%$). The divergence classifier improves the diversity between task tokens: we observed that the minimal Euclidean distance between them increases by 8%. Moreover, we also remarked that having independent classifiers reduces the Chaudhry et al.'s forgetting [7] by more than 24%.

5. Conclusion

In this paper, we propose DyTox, a new dynamic strategy for continual learning based on transformer architecture. In our model, self-attention layers are shared across all tasks, and we add task-specific tokens to achieve task-specialized embeddings through a new task-attention layer. This architecture allows us to dynamically process new tasks with very little memory overhead and does not require complex hyperparameter tuning. Our experiments show that our framework scales to large datasets like ImageNet1k with state-of-the-art performances. Moreover, when a large number of tasks is considered (*i.e.* CIFAR100 50 steps) our number of parameters increases reasonably contrary to previous dynamic strategies.

Limitations: True continual learning aims at learning an almost unlimited number of tasks with low forgetting. No current approaches are yet able to do so. Thus, forgetting is not yet solved for continual learning but our model is a step forward in that direction.

Broader impact: Machine learning models often are biased, with some classes suffering from lower performances. Studying forgetting in continual learning provides insights about the difference in performances between classes. Our task-specialized model could help reduce these biases.

Acknowledgments: This work was partly supported by ANR grant VISA DEEP (ANR-20-CHIA-0022), and the HPC resources of IDRIS AD011011706.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2018. (page 2).

[2] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey Hinton. Layer normalization. In *Advances in NeurIPS 2016 Deep Learning Symposium*, 2016. (page 3).

[3] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. (page 4).

[4] Francisco M. Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2018. (pages 2, 6, 8).

[5] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 2).

[6] Olivier Chapelle, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001. (page 7).

[7] Arslan Chaudhry, Puneet Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2018. (pages 2, 8).

[8] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. (page 2).

[9] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H.S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. In *International Conference on Machine Learning (ICML) Workshop*, 2019. (page 2).

[10] Mark Patrick Collier, Effrosyni Kokiopoulou, Andrea Gesmundo, and Jesse Berent. Routing networks with co-training for continual learning. In *icmlws*, 2020. (page 2).

[11] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Giulio Morcos, Ari annd Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *arXiv preprint library*, 2021. (pages 2, 5).

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. (page 5).

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018. (pages 2, 5).

[14] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 2).

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. (pages 2, 3, 4, 5).

[16] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (page 2).

[17] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Tackling catastrophic forgetting and background shift in continual semantic segmentation. In *arXiv preprint library*, 2021. (page 2).

[18] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2020. (pages 1, 2, 5, 7, 8).

[19] Arthur Douillard and Timothée Lesort. Continuum: Simple management of complex continual learning scenarios. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2021. (page 6).

[20] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Internaltional Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020. (page 2).

[21] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv preprint library*, 2017. (pages 1, 2).

[22] Robert French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999. (page 2).

[23] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, 2017. (page 3).

[24] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, 2019. (pages 1, 2).

[25] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017. (page 7).

[26] Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2020. (page 2).

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 2).

[28] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, 2015. (pages 5, 8).

[29] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (pages 1, 2, 5, 6, 7, 8).

[30] Steven C.Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (pages 1, 2).

[31] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2020. (page 2).

[32] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver io: A general architecture for structured inputs outputs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. (page 2).

[33] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. (page 2).

[34] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017. (pages 1, 2).

[35] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009. (page 5).

[36] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, 1999. (page 2).

[37] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. In *arXiv preprint library*, 2015. (page 2).

[38] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *International Joint Conference on Neural Networks*, 2019. (page 2).

[39] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. (pages 1, 2).

[40] Z. Li and D. Hoiem. Learning without forgetting. *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2016. (page 2).

[41] Zhuoyun Li, Changhong Zhong, Sijia Liu, Ruixuan Wang, and Wei-Shi Zheng. Preserving earlier knowledge in continual learning with the help of all previous feature extractors. In *arXiv preprint library*, 2021. (pages 1, 2, 6).

[42] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. (page 2).

[43] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (page 2).

[44] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014. (page 5).

[45] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. (page 2).

[46] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*, 2019. (page 5).

[47] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Ming-Hsuan Yang. An adaptive random path selection approach for incremental learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (pages 6, 7).

[48] Dhanesh Ramachandram and Graham W. Taylor. Deep multimodal learning: A survey on recent advances and trends. In *IEEE Signal Processing Magazine*, 2017. (page 4).

[49] 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 (CVPR)*, 2017. (pages 1, 2, 5, 6, 7).

[50] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995. (page 2).

[51] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint library*, 2016. (page 2).

[52] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. (page 2).

[53] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from

deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 5).

[54] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018. (pages 1, 2).

[55] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (page 2).

[56] Sebastian Thrun. Lifelong learning algorithms. In *Springer Learning to Learn*, 1998. (page 2).

[57] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. (page 7).

[58] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021. (pages 2, 5, 6).

[59] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. (pages 2, 4, 5).

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (pages 2, 3, 4, 5).

[61] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: An alternative approach to efficient ensemble and lifelong learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. (page 2).

[62] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. (page 6).

[63] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (pages 1, 4, 7).

[64] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. (pages 1, 2, 5, 6, 7, 8).

[65] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. (pages 1, 2).

[66] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 2017. (page 2).

[67] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. (page 7).

[68] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shutao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (pages 4, 5, 6, 7).

[69] Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S. Davis. M2kd: Multi-model and multi-level knowledge distillation for incremental learning. *arXiv preprint library*, 2019. (page 2).